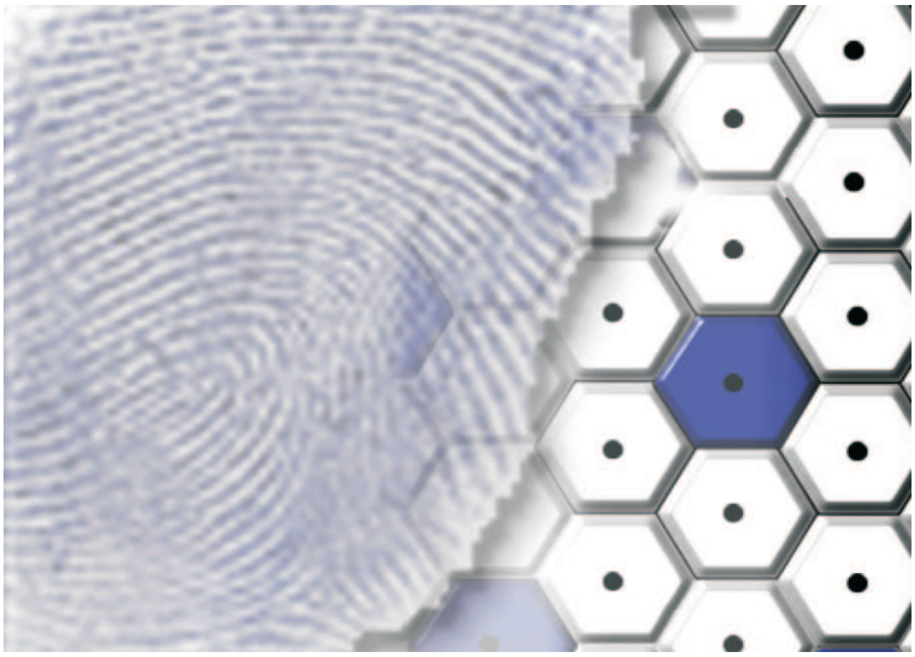
Prof. Dr.-Ing. Firoz Kaderali

Network Security



Author

Prof. Dr.-Ing. Firoz Kaderali

1963 - 69	Studied theoretical electrical engineering at the Technische Hochschule Darmstadt
1969 - 74	Assistant at the Faculty of Electrical Engineering at the Technische Hochschule Darmstadt
1974	Doctorate (Promotion) at the Faculty of Electrical Engineering, Technische Hochschule Darmstadt, Subject: Network Theory
1974 - 76	Dozent für Statistische Signaltheorie at the Technischen Hochschule Darmstadt
1976 - 81	Member of the Research Center at SEL (ITT)/Stuttgart. Projektleader of Bundespost study and Fieldtrial DIGON (Digital Local Area Network)
1981 - 86	Head of the Department System Development/Large Systems at Telefonbau und Normalzeit/Frankfurt Development of ISDN-PABXs
Since 1986	Professor for Communications Systems at the FernUniversität Hagen Main interests: Communications Systems, Networks and Protocols; Network Security
1989 - 94	Head of the regional telecommunications initiative TELETECH NRW (Consultant and supervisor of over 120 Telecommunications Projects)
1990 - 96	Member of the ISDN Research Commission NRW (Projects on ISDN Applicationsdesign and Technology Assessment)
Since 1992	Director of the Research Institute of Telecommunications (FTK) in Dortmund, Hagen and Wuppertal, Joint Institute of the Universities Hagen (Electrical Engineering) and Wuppertal (Economy)
1995 - 2001	Member of management of mediaNRW, an initiative to promote the development and spreading of multimedia-applications as a whole and interactive services in enterprises, private households, and the public sector
1999 - 2003	Chairman of the research alliance Data Security Northrhein-Westphalia
2000 - 2002	Chairman of the advisory board of GITS (Gesellschaft für IT-Sicherheit) in Bochum, Germany
Since 2002	Chairman of the open source initiative Campus Source

Table of Contents

1 Introduction

1.1 IT-Security in a networked world

The Internet is a world-wide network of computers, which has grown historically. The resulting work space (Cyberspace) is full of dangers, since protection from attacks on computers and their communication is incomplete. This is essentially due to the fact that during the development of the language of the Internet, the Transport Control Protocol/Internet Protocol (TCP/IP), security aspects were not taken in to consideration. For example TCP/IP in the wide-spread version 4 does not know of encryption (not even the encryption of passwords). Furthermore the sender addresses of a message can easily be forged and even complete messages can be forged or redirected by the intermediate nodes of the network, the routers. When using the Internet for both private and commercial applications it is therefore necessary to take additional security measures.

Cyberspace

TCP/IP

If an unauthorized person gains access to a computer or a communication network, data is in danger of being spied on, forged or deleted. Even the computer or network itself can be tampered with or crash. Depending on the application affected, an attack can have diverse and sometimes even disastrous consequences. Attacks range from industrial espionage and spying on public offices to the forgery and prevention of business and financial transactions. Personal privacy can also be affected.

The problems faced by Internet Security which are presented here, are only one aspect of Network Security dealt with in this course. Similar security problems can, in principle, be identified in every network and protocol, both wired and wireless. The number and scale of these IT-Systems has been constantly growing for several years and as a result of that, IT-Security has become more and more significant.

This becomes clear when the number and variety of attacks on network security is regarded. The following list provides a small selection of incidences:

I Love You *Virus, Mai 2000*
Estimated damage: 15 Bil. $

Code Red *Worm, July 2001*
Self-replicating malicious code that exploits a known vulnerability in Microsoft IIS servers. Performs Distributed Denial of Service (DDoS) attack on www1.whitehouse.gov.

Beagle *mass-mailing Worm, January 2004*
Distributes by using its own SMTP engine and over P2P-Networks. Works as Spam-Relay. Deactivates Virus-Scanners and other security related software.

Phatbot *Worm/Botnet, April 2004*
Distributes by using backdoors of other Worms and Viruses as well as a wide range of exploits and via mass-mailing. Builds remote controllable P2P net-

works for e.g. DDoS attacks and SPAM actions. ”Features” could be extended remotely on a modular basis.

EPOC.Cabir *Virus, June 2004*
First Bluetooth-Virus.

Spoofing attack *July 2004*
on German credit institute “Postbank”.

This list represents only a small excerpt of current incidents. Further occurrences can be found almost daily in the press. The large number of successful attacks on data and network security does not equate with the amount of technical security measures currently available. An analysis of threating scenarios leads to the following table:

Threats	Security Measures
Eavesdropping	Encryption
Falsifying of Messages	Authentication of Messages (Digital Signatures)
Wrong Personalization	Authentication of Persons
Observing	Anonymity Techniques (e.g. Mixes)
Copyright Infringement	Digital Watermarks
Viruses, Worms, Trojans, Spoofing	Virus-Scanners, Firewalls, Intrusion Detection
Ad-ware	Ad- and Malware-Scanner
Spam	Spam-Filters

A detailed analysis of these technical security measures (which are mostly based on hard cryptological methods) would go beyond the scope of this introduction. However from the author’s opinion, these methods are adequate to provide reasonable data and network security. In practice however the use of these methods is insufficient. This again is on the one hand because of the lack of user know-how and technical personnel (the lack of perception of dangers also belongs to this) and on the other hand because of the lack of suitability of these products and service for everyday life. With respect to the scope of network security, significance should be placed on education as well as on research and development.

In this course unit the application of above mentioned security mesures in current networks and protocols is elaborated in more detail.

1.2 Overview of this course

The Course *Network Security* consists of two parts: *Network Security I* and *Network Security II*. Each part consists of seven course units. The seven course units of Network Security I cover the six chapters: Introduction, Basics, Internet Security Protocols, World Wide Web Security, Anonymity Techniques, Packet Filters and Firewall Systems. Network Security II covers six chapters: Application Layer Security, Security in Wireless and Mobile Networks, Electronic Payment Systems, Security Aspects in Mobile Agent Systems, Copyright Protection and Intrusion Detection.

The following sections provide an overview of the contents of Chapters 2-12.

This chapter closes with a list of references and recommended readings.

1.3 Basics

In order to deal with network security, some basic knowledge is required. The kinds of attackers and attack scenarios on hosts and networks must be known as well as the security models and levels which can be extrapolated from them.

Furthermore a fundamental understanding of the basic techniques of communication networks is necessary. Knowledge and an understanding of the seven layers of the OSI reference model, the utilized switching techniques and the components involved in communication networks is of fundamental importance.

Additionally one should be familiar with the internet protocol family, should be familiar with the emergence and the history of the internet as well as the structure of the internet protocol stack. The fundamental protocols of the internet layer (IP) and the transport layer (TCP, UDP and ICMP) are of significant importance along with the security problems of the IP protocol family and common internet services.

An outline of this basic knowledge is given in Chapter 2.

1.4 Internet Security Protocols

The Internet protocol suite (TCP/IP) has been designed without consideration of security. It is very easy to spoof IP addresses, manipulate DNS and to eavesdrop the links.

Recently several cryptographic protocols have been proposed, specified, and partly implemented in the Internet and the WWW. The development of Internet standards for security (i.e. RFCs[2] published as a standard track document) has been slow. The first RFC to introduce a security-oriented protocol into the TCP/IP suite did not appear until 1987, with the publication of the first e-mail security protocol specifications. In the following years, a number of security-oriented Internet protocols

2 Request for Comment

were developed and are recently at various stages in the Internet standardisation process. Internet standards are formally developed under the auspices of the **Internet Society**, whose technical arm is the **Internet Architecture Board (IAB)**. IAB consists of two taskforces: **Internet Research Task Force (IRTF)** and **Internet Engineering Task Force (IETF)**. IETF consists of a large number of working groups (WG), where the bulk of standard development takes place. Despite the greatly increased activity in security standards in the IETF, the Internet is still not completely secure (the protocols developed must be used!).

Internet Society, IAB, IETF, IRTF

In chapter 3 we outline and briefly discuss some Internet security protocols. In the case of TCP/IP based networks, cryptographic security protocols can operate at any layer of the corresponding communications protocol suite. There are many proposals for providing security services at the network access, IP, transport and application layer. Here we focus on the **IPSec** set of security protocols in the IP layer and two transport layer security protocols, namely the **secure sockets layer (SSL)** and the **transport layer security (TLS) protocols**.

IPSec

SSL, TSL

1.5 World Wide Web Security

The World Wide Web was originally developed as a publishing medium for public documents and therefore provided few controls for restricting access to information. As a wider range of documents and services appeared on the web, improved security facilities to satisfy the new requirements were needed.

There are basically three overlapping types of security risks regarding the World Wide Web ([Ste98]):

1. Bugs or misconfiguration problems in the Web server.
2. Browser-side risks.
3. Interception of network data sent from browser to server or vice versa by network eavesdropping.

Following a short introduction, chapter 4 deals with the principles of authentication in the sessionless **Hypertext Transfer Protocol (HTTP)** and then in Section 4.3 “Server-side security” and Section 4.4 “Client-side security” with the first two of the above-mentioned risks. Measures to protect from the third risk are dealt with in Chapter 3.

Hypertext Transfer Protocol (HTTP)

1.6 Anonymity Techniques

The objective of encryption is to achieve confidentiality of information. However, the fact that two people communicate is not disguised. The frequent opinion *Those who do not have to hide something need not be anonymous* may apply to everyday life but is not relevant in communication networks like the Internet, where vast possibilities for data recording and data mining are available.

For example, each user leaves traces when an e-mail is sent: all computers involved in the transport know both the sender and the receiver of the message. If this – as in most cases – is not encrypted, the content can be observed. This also applies to other services such as FTP (File Transfer Protocol), chat sessions or newsgroups.

There are examples of everyday life where not only the content of a message but also sender, receiver, and the fact that they communicated, are anonymous:

- Paying cash is an anonymous business. The payer cannot be identified via the coins used.
- Phone calls to charitable organizations (Samaritans, help for drug addicts etc) have to be anonymous. Such a guarantee is also desired by anonymous newsgroups in electronic networks.
- Box numbers in newspapers: the name of the person who advertised remains unknown.

If we transfer these examples into electronic communications environments, it is often difficult to satisfy both the demand for anonymity and reliability. Many solutions for the problems mentioned, do not achieve complete anonymity, they only achieve pseudo anonymity where a trustworthy third party carries out the anonymization.

Mix-concept, DC-concept

Chapter 5 deals with some basic ideas of anonymity used in the **Mix-** and the **DC-concept** and introduces the following four basic types of anonymity: Sender Anonymity, Recipient Anonymity, Mutual Anonymity and Unobservability.

1.7 Packet Filters and Firewall Systems

With host security you enforce the security of each host machine separately and make the effort to avoid or alleviate all the known security problems that might affect that particular host. Host security is hard to achieve and does not scale in the sense that as the number of hosts increases, the ability to ensure that security is at a high level for each host decreases. On the other hand, a network security model concentrates on controlling network access to your various hosts and the services they offer, rather than on securing them one by one. Network security approaches include building intermediate systems to protect your internal systems and networks, using strong authentification approaches, like public-key or one-time passwords, and using encryption and integrity checks to protect particularly sensitive data as it transits the network through routers.

Packet Filters
Firewall Systems

In Chapter 6, we discuss possible solutions to achieve network security with **packet filters** and/or **firewalls (firewall systems)**. There are many different definitions of the term ”firewall” in the literature. A firewall represents a blockade between a privately owned and protected intranet, that is assumed to be secure and its users are trusted, and another network, typically a publicly owned network or the Internet. The later is assumed not to be secure and not trustworthy. The purpose of the firewall is to prevent unwanted and unauthorized communication into or out of the protected network.

In [CB94] a firewall system is defined as a collection of components placed between two networks that collectively have the following properties:

1. All traffic from inside to outside, and vice versa, must pass through the firewall.
2. Only authorized traffic, as defined by the local security policy, will be allowed to pass.
3. The firewall system itself is immune to penetration.

These properties are design goals. A failure in one aspect does not necessarily mean that the collection is not a firewall, but simply that it is not a good one. Consequently, there are different grades of security that a firewall can achieve. Note that there must be a local security policy when the rules of the firewall systems are established.

Application Gateways

Other definitions of firewall systems include connections and data streams from inside to outside and vice versa and must be strongly authenticated by users. In this case, the firewall system has to operate at higher layers in a communication protocol stack where information about users is available. These systems require the use of **application gateways**. Systems which only operate at lower layers without authentification information about the end users or connections are called packet filters. For the sake of clarity, in Chapter 6 we make a clear distinction between packet filters (operating at the network layer, Internet layer or transport layer in the Internet protocol stack) and firewalls operating at some higher layer.

The contents of Chapter 6 are categorized in five parts. Beginning with part 1 which deals with packet filtering followed by part 2 and 3 which introduce circuit-level and application-level gateways. Part 4 discusses how to set up network topologies for packet filters or firewall systems to make an intranet secure and finally, part 5 lists some further reading.

1.8 Application Layer Security

Users interact at the application layer of the OSI reference model for communication. The application layer must be protected as vigorously as the other layers of the OSI model.

Providing security at the application layer is also the most flexible, because the scope and strength of the protection can be tailored to meet the specific needs of the application [Opp00]. With this approach, application protocols must be modified to provide security services as well as programs written for these protocols. A disadvantage is that these new applications must be available to both communication parties. For example, S-HTTP (Secure Hypertext Transfer Protocol) is a security-enhanced HTTP protocol. An S-HTTP session requires the availability of an S-HTTP-capable client and an S-HTTP-capable server.

The focus of Chapter 7 is on security issues intrinsic to the application layer.

There are many application protocols and services layered on top of TCP and UDP (User Datagram Protocol). The most important of them will be outlined next. For each application protocol, we will focus on its security weaknesses and give alternatives, or security-enhancements proposed to provide the security required. So in Chapter 7, we will deal with:

1. Remote terminal access, implemented by the Telnet remote login protocol.
2. File transfer, implemented by the file transfer protocol (FTP).
3. The network file system (NFS) which uses RPC (Remote Procedure Call) to provide transparent file access over a network.

The topics S/MIME and executable content which are subject to network security will be introduced in Chapter 7.

1.9 Security in Wireless and Mobile Networks

Mobile networks have become a very attractive channel for the provision of electronic services: They are available almost anytime, anywhere, and the user acceptance of mobile devices is high. As a result, there is an ever increasing amount of services offered by mobile networks. They range from simple speech and information services to sensitive applications like banking or electronic commerce. But this is not the only reason that the security of data and signalling of information play a very important role. Communication traffic can be eavesdropped by everyone with very simple devices due to the very sensitive radio path of the data.

GSM, UMTS, WAP, WLAN, Bluetooth

In Chapter 8 we will a have brief look at the development of mobile networks to date. The main discourse in Chapter 8 concentrates on the security in **GSM** networks. Further topics are the security mechanisms in **UMTS** networks, in the **WAP** protocol, in **WLAN** and in the **Bluetooth** standard.

1.10 Electronic Payment Systems

With the rapid growth of the Internet, and particularly the World Wide Web (WWW), a major new industry has developed worldwide – electronic commerce. Online auctions, shopping portals and online book stores have become routine in our daily life. Also in areas of communication and transactions between government and public authorities with citizens (so-called **G2C**) and in **business to business (B2B)** the Internet and the WWW are used for more efficiency and for faster and more comfortable completion of real world processes. But whenever transactions are involved, in most applications payment is made in traditional ways. The most common payment method used in the WWW today is Credit-Cards.

G2C
B2B

Electronic payment systems enable secure payment (transfer of funds between different parties) in insecure network environments using newly developed cryptographic techniques. Most of us are familiar with electronic payment: we check our account balance and tenants transfer rent, gas, water and electricity bills via on-line internet services, etc. Electronic payment systems can be characterized in several ways: by the way in which money transfer is organized or by the type of information to be exchanged. Existing payment systems are Credit Card-based, electronic check, electronic cash and micropayment systems. Furthermore, electronic cash systems may be distinguished as **on-line** or **off-line systems** according to whether banks are involved during the payment process. Electronic cash resembles conventional cash. In Credit Card-based systems, bank accounts of customers are transferred via open networks and money is represented by numbers in the accounts. The Micropayment system is a special group in which the value of money per transaction is small and fixed with lower security requirements.

on-line, off-line systems

Various secure network payment schemes have been developed at universities and different research institutes as well as commercial organizations. Some of them have undergone small scale testing and some of them have been proven to fail for some reasons (for example, some unconditional privacy protecting systems could be eventually misused by criminals for blackmailing or money laundering).

New technologies, including new security tools, new cryptographic algorithms and new protocols are needed to protect privacy during transactions and to make the systems more secure, more efficient and more acceptable to organizations and individuals.

Chapter 9 introduces the main technologies involved in most payment systems currently available to network users. The characteristics of electronic payment systems are described in Section 9.2. We classify electronic payment systems in Section 9.3 and explain some systems from each category in Section 9.4. Chapter 9 ends with a bird’s eye view on the future of electronic payment systems.

1.11 Security Aspects in Mobile Agent Systems

Over the last decade, computer systems have evolved from centralized computing devices supporting static applications, into client-server environments that allow complex forms of distributed computing. Throughout this evolution limited forms of code mobility have existed: the earliest being remote job entry terminals used to submit programs to a central computer and the latest being Java applets downloaded from web servers into web browsers.

A new phase of evolution is now under way that goes one step further, allowing complete mobility of cooperating applications among supporting platforms to form a large-scale distributed system. This evolutionary path is the **mobile agent technology**. The mobile agent technology offers a new computing paradigm in which a program in the form of a software agent or a mobile agent, can start the execution of its code on a host computer, transfer itself to another agent-enabled host on the network, and resume the execution of the code on the new host.

mobile agent technology

The challenges of mobile agents lie in the lack of proven applications, security, infrastructure and standards. Applications using mobile agent technology are, for example e-Commerce, Software distribution, Information retrieval, System administration and Network management. Infrastructures can be viewed as system components. In a mobile agent system, these infrastructures (i.e. communication, naming ser- vice, controlling and locating) should be provided. Communication defines how a mobile agent communicates with other mobile agents. Some existing commu- nication protocols are message passing protocol and synchronous communication. Naming service is the process of efficiently naming a mobile agent in such a way that each mobile agent possesses a unique name. In other words, given a mobile agent's name, one can identify the owner of this mobile agent and distinguish this mobile agent from other mobile agents belonging to the same owner. Locating is the process of locating the current position of a mobile agent in the system. Controlling is composed of how mobile agents migrate (i.e. code serialization), how a visited host executes a visiting mobile agent.

Standards characterize general rules for applications using mobile agent implementations. Security is hard to achieve for mobile agents. It is generally composed of protecting mobile agents from malicious hosts and protecting hosts from malicious mobile agents. In Chapter 10, we emphasize security issues of mobile agent systems. The contents of Chapter 10 are categorized into four parts. Beginning with part 1 which provides an introduction to the agent technology. Part 2 demonstrates the history of the mobile agent concept. Part 3 introduces an overview of mobile agent systems. And finally, part 4 explains explicitly security issues of mobile agent systems.

1.12 Copyright Protection

Not everything on the Internet is public domain and may be taken without permission from the creator/owner. Copyright is the protection of published works. There are some technologies that can be used to protect and enforce copyrights on works published on the Internet. Two of these technologies are **watermarking** and **fingerprinting**.

watermarking, fingerprinting

Copyright protection becomes more difficult in the digital world than in the analogue world. A copy of digital data can be easy, inexpensive, and quickly created and distributed. Moreover, such a copy is not distinguishable from the original[3].

In Chapter 11 we will a have brief look at Copyright protection.

1.13 Intrusion Detection

With an increasing number of interconnected devices, modern technology can be used to it´s fullest potential, but, at the same time the potential risk of attacks which can be insider misuse, theft of proprietary information, viruses, software vulnerabilities, etc. also rises. With regard to vulnerabilities reported to the Computer Emergency Response Team (CERT), there was an increase of a factor of 32 in the past ten years (http://www.cert.org). Further, it can be observed that the attacks are also becoming more sophisticated – from password guessing at the beginning to more advanced attacks today such as malicious code attacks, website defacement, Distributed Denial of Service (DDoS), etc.

The Internet is an open and distributed system and a security challenge, especially as this open system becomes more complex. This, along with growing interconnectivity[4], the complexity of applications, operating systems and related protocols offered via the network as well as subconscious security behaviour of many users have led to a growth in security threats. One of the measures used to fight against these security threats is Intrusion Detection.

The idea behind the technique of detecting intruders is that when an intruder gets into a third party system, he leaves traces behind or behaves differently compared to the normal user. This means that if methods or mechanisms can be found to carefully analyse the dataproduced, there is a greater chance of detecting a violation of the system policy and hence the intruders. Depending on how the data is collected and analysed we speak of **Host-based Intrusion Detection Systems (HIDS)**, where the software is installed on a single host or of **Network based Intrusion Detection System (NIDS)**, whereby the IDS Software monitors a network segment or a complete network. To detect attacks on information systems, basically two approaches can be distinguished:

HIDS

NIDS

Anomaly Detection

- **Anomaly Detection** (Statistical Approaches, Bayesian Networks, Neural Networks, etc.) and

Misuse Detection

- **Misuse Detection** (Pattern Matching, State Transition Analysis, etc.).

3 In the analog world, as the number of copies increases, the quality of these copies decreases.

4 The number of Internet hosts increased from 4.852.000 in January 1995 to 394.991.609 in January 2006 (http://www.isc.org)

In Anomaly Detection a profile of the variable to be analysed is made e.g. the use of resources (CPU usage) or typical user behaviour (login behaviour, execution order and frequency of programs, etc.), this is also called the long term profile and is compared with the actual realisation of the variable (short term profile). If the difference between long term and short term behaviour exceeds an a priori defined threshold, the event can then be seen as an anomaly [Den87].

A different approach is followed in Misuse Detection whereby attack specific signatures are stored in the signature database. The data stream is then systematically analysed by searching for these attack signatures in the data stream. If a match occurs, there is an intrusion.

In the first part of Chapter 12, the basic principles which are primordial for the understanding of intrusions are introduced. We then present a brief scenario on how intrusions are prepared and subsequently different types of intrusions (protocol related intrusions, remote access intrusions, Malicious code, etc.) are dealt with. In the second part we show why traditional security mechanisms such as Access Control Lists (ACLs) and Firewalls fail to solve the intrusion related problems we are facing today. In closing, we deal with Intrusion Detection, introduce concepts and investigate methods and where required, related techniques in greater detail.

1.14 Recommended literature for this course

There are many good reference books for communication techniques and cryptography. Here we mention some books which can be useful for further study.

For the general topic of communication techniques: [Kad91], [Kad95], [KDS+00], [Tan00] and [Spu00].

For the special topic on Internet protocols and services: [Los99], [Fei99], [Com00] and [KDS+00].

For a general understanding of cryptographic definitions, algorithms and protocols we recommend the following books and courses: [MOV96], [Sti95], [Sch96], [Buc99] and [KCL+00].

For the special topic on network and Internet security: [Smi97], [Fuh98], [FRU00], [Opp00], [DH99] and [Eck06].

2 Basics

2.1 Host and network security

Internet and mobile networks, like GSM (Global System for Mobile Communications) and UMTS (Universal Mobile Telecommunications System), are marvelous advanced technologies that provide access to information, and the ability to publish information, in revolutionary ways. But there is also a major danger that provides the ability to destroy information or to gain illegitimate access to services of networks or computers, which are connected through this networks. The beforementioned network technologies are widely used not only for personal use, but also for confidential exchanges, like purchase, contracts and electronic payments transactions, between **business and consumers (B2C)**, **business and business (B2B)** and **governments and their citizen (G2C)**.

B2C, B2B, G2C

This course describes on the one side the risks of currently used protocols and network services and on the other side methods to make network protocols and services more secure with the use of cryptographic primitives and protocols against active or passive attackers. Under active attacks we understand that an attacker intercepts, changes or deletes protocol messages or he gets unauthorized control over a computer or a resource.

2.1.1 Types of attacks

Suppose a computer is connected to the Internet. The data and the resources of the computer and the communication between the computer with others can be object of passive or active attacks. The computer or network administrator must on the one hand protect the data and resources of computers from non-authorized persons. And on the other hand secure protocols have to be used to access information and resources of the computer for authorized persons on remote computers. The following characteristics of the data on the computer have to be protected: access, secrecy and integrity. For messages of network protocols secrecy, integrity and authenticity should be guaranteed.

The attacks on systems and networks can be categorised into the following three **basic types of attacks** ([ZCC00]):

basic types of attacks

1. Intrusion

   The most common attacks on systems are intrusions, people want be able to use your computer or resources as if they are legitimate users on your system. There are a dozens of ways to get access for an attacker. They range from social engineering attacks, guessing login names and passwords, to intercepting password based authentification protocols or just finding a hole in the system, where no authentication is needed. Secure authentication methods with passwords, firewalls and intrusion detection systems are ways to block, log or control these types of attacks.

2. Denial of service

   A denial of service attack is one that is aimed entirely at preventing you from using your own computers or network services. These attacks are launched by flooding a network or a computer with messages, so that regular messages to or from your computers have no chance to reach their destination. An intruder floods a system or network with messages, processes, or network requests so that no real work can be done. The system or network spends all its time responding to messages and requests, and can't satisfy any of them. Flooding is the simplest and most common way to carry out denial of service attacks, but one can also disable services, reroute them, or replace them.

   It is close to impossible to supress all denial of service attacks. Whenever you want some services like electronic mail or remote login on your system, they can be flooded. A good security shield is to secure your intranet with a firewall system separating your local area network from the Internet. Flooding attacks are considered unsporting by many attackers, because they are not very difficult to carry out. In most cases flooding attacks are pointless, because no secret information is derived.

3. Information theft

   Many types of attacks allow an attacker to get data without ever having to directly use your computer. Usually these attacks exploit Internet services or network protocols that are intended to give out information, inducing the services to give out more information than was intended, or to give it to the wrong people. These services include the WWW service via the HTTP (Hypertext Transfer Protocol) protocol, the FTP (File Transfer Protocol), the finger service, and many more. Many Internet services like Samba or NFS (Network File System) are designed for use on local area networks, and do not have the type or degree of security that would allow them to be used safely across the Internet. Information theft does not need to be active or particularly technical.

2.1.2 Types of attackers

types of attackers

Next, we will describe the **types of attackers** you find on the Internet. There are many ways to categorize these attackers ([ZCC00]). All attackers share certain characteristics. They do not want to be caught, so they try to tarn themselves, conceal their identity and their real geographic location. If they gain access to your system, they will certainly attempt to preserve that access, if possible, by building in additional ways to get access. Also they can use your computer to launch attacks on other sites to conceal their real geographic location.

1. Joyriders

   Joyriders are bored people looking for amusement. They break in because they think you might have interesting data, or because it would be amusing to use your computer, or because they have nothing better to do. They are curious but not actively malicious. Joyriders are particularly attracted to well-known sites and uncommon computers.

2. Vandals

   Vandals are out to do damage, either because they get their kick from destroying things, or because they do not like you. When a vandal gets access to your network or computer, you will notice it. Fortunately, vandals are fairly rare. Most of them go for straightforward destruction, which is unpleasant but relatively easily detected and repaired. Im most circumstances, deleting your data, or even ruining your computer equipment, is not the worst thing one could do to you, but is what vandals do. Unfortunately, it is that nearly impossible to stop a determined vandal. Certain attacks are particularly attractive to vandals but not to other types of attackers. For example, denial of service attacks are not attractive to joyriders, while joyriders are around in your system, they are just as interested as you are in having your computers, running, and available to the Internet.

3. Scorekeepers

   Many intruders follow in an updated version of an ancient tradition. They collect merits, based on the number and types of systems they break into. Like joyriders and vandals, scorekeepers may prefer sites of particular interest. Breaking into something well known and well defended, is usually worth more points to them. However, they also attack anything they can get at. They go for quantity as well as quality. They will certainly gather valuable information and keep it for later use. They will probably try to leave themselves ways to get back in later and will use your machines as a platform to attack others. Many scorekeepers are not technical experts but use programs and scripts written by other people and follow instructions on how to use them.

4. Spies

   Some attackers break into computers to get information, that can be directly converted into money or further access, e.g. credit card numbers, or network access information. As far as known, serious computer-based espionage is much rarer, outside of traditional espionage circles. Espionage is much more difficult to detect than break-ins. An information theft need not leave any traces at all, and even intrusions are relatively rarely detected immediately. Good spies break in, copy data and leave without disturbing anything.

   In practical terms, most organisations can not prevent spies from succeeding. The precautions that governments and firms take to protect sensitive data on computers are complex and expensive. The precautions include electromagnetic shielding, careful access control, and absolutely no connections to unsecured or open networks.

Much of security is about trust in software implementations, hardware and persons. The question is: ”Who do you trust to do what?” The world does not work unless you trust some people to do some things, and security people sometimes seem to take an overly suspicious attitude, trusting nobody. Why shouldn’t you trust your users, or well known software vendors?

2.1.3 Security models

After having outlined basic attack scenarios on computers and networks, we will now describe some **security models** and levels to protect your machines:

security models

1. No Security

   The simplest possible approach is to put no effort at all into security, and run whatever minimal security your software and hardware vendors provide you as default. In this model anything that is possible is allowed to everybody. This security model was also introduced in former releases of Linux distributions: All possible network services were enabled. The default settings in newer Linux distributions enable only a few network services as default.

2. Security through obscurity

   Another possible security model is the one commonly referred to as security through obscurity. In this model, a system is assumed to be secure simply no relevant information about the security model is available - its existence, contents, security measures, or anything else. This approach seldom works for long and is not recommended.

3. Host security

   Probably the most common model for computer security is host security. In this model you enforce the security of each host machine separately and make the effort to avoid or alleviate all known security problems that might affect that particular host.

   The major impediment of effective host security in modern computing environments is the complexity and the diversity of these environments. Most environments include machines from multiple vendors, each with its own operating system, and each with its own set of security problems. Even if the site has machines from one vendor, different releases of the same operating system often have significantly different security problems. Even if all these machines are from a single vendor and run a single release of the same operating system different system configurations can bring different subsystems into play and lead to different sets of security problems. The sheer number of machines at some sites can make securing them all difficult.

   It takes a significant amount of ongoing work to effectively implement and maintain host security. Even if all the work is done correctly, host security still often fails due to bugs in vendor software, or due to a lack of suitably secure software for some required functions.

Host security also relies on the good intentions and the skill of everyone who has privileged access to any machine. As the number of machines increases, the number of privileged users increases as well. Securing a machine is much more difficult than attaching a maschine to a network, so insecure machines appear on your network as unexpected surprises.

A host security model may be highly appropriate for small sites with extreme security requirements. Indeed, all sites should include some level of host security in their overall security plan. Even if you adopt a network security model as we describe below, certain system components in the configuration of a machine will benefit from host security.

4. Network security

   As environments grow larger and more diverse and as securing them on a host-by-host basis grows more difficult, more sites turn to a network security model. A network security model concentrates on controlling network access to various hosts and the services they offer, rather than on securing them one by one. Network security approaches include building firewalls to protect your internal systems and networks, using strong authentication approaches, like public-key or one-time passwords, and using encryption and integrity checks to protect particularly sensitive data as it transits the network through routers.

   A site can get tremendous leverage from its security efforts by using a network security model. For example, a single network firewall of the type we discuss in chapter 6 can protect hundreds, or even thousands of machines against attacks from networks beyond the firewall, regardless of the level of host security of individual machines at the site. The network security model is a necessary, but not a sufficient approach to get overall security. Also the protocols and services, which are visible and accessible from outside your network must be configured well and include security mechanisms.

   This kind of leverage depends on the ability to control the access points to the network. At sites that are very large or widely distributed, it may be impossible for one group of people to even identify all of the access points, much less control them. At this point, the network security model is no longer sufficient, and it is necessary to use layered security, combining a variety of different security approaches.

No security model can solve all problems. No security model can prevent a hostile person with legitimate access from purposefully damaging your site or taking confidential information out of it. And, no security model can take care of management problems.

In the next section (Section 2.2) we describe basic communication techniques and the basic communication model for network protocols: OSI (Open System Interconnection) of the ISO (International Standardization of Organizations). This model describes a layered model for communication processes. The most widely used protocol is the IP protocol family with its services. Many network services are build and implemented via the IP protocol. We will describe the basic facts about the IP protocol in Section 2.3. This enables you to analyse the security risks of the IP protocol and its services and to understand and build secure protocols over IP.

2.2 Basic techniques in communication networks

A network consists of nodes and links. Nodes are abstract representations of communication and transmission equipment. In thess nodes a subset of communication functions is implemented and may be separated into layers as explained below. Links are physical transmission facilities, e.g. copper or fibre optical cables, to interconnect nodes. The simplest form of a computer network consists of two nodes connected by a single link. The communication in networks is controlled by a complex set of rules, the protocols.

In the 1970's, a number of companies and university departments developed computer networks. Each company used a different structure or architecture for its networks, as there are many different ways in which network functions can be organized. Despite their differences, the various architectures used in these early networks were all organised in to layers. Introducing a layered model always means to group related functions together and implement communications software in a modular manner. Such a group of related communication functions is called a layer of a communication model.

ISO 7498-1, X.200

In the late 1970'ies, ISO proposed an architecture model (I**SO 7498-1** and later as **ITU-T recommendation X.200** (International Telecommunication Union)) called the Open System Interconnection (OSI) model ([ISO84]). The OSI model is a layered architecture dividing network functions into seven conceptual layers. The network functions should realise the behaviour of a protocol.

The OSI model was an international effort to create standards for computer and generic application services. The OSI reference model permits the interconnection of systems of different origins at different layers of this model. The OSI model is not concerned with the internal architecture of systems but with their external behaviour. Seven standardised layers correspond to two groups of functions: The transmission oriented layers and the application oriented layers (see Fig. 2.2-1).

Layer	Group
7 Application Layer	application oriented layers
6 Presentation Layer	
5 Session Layer	
4 Transport Layer	transmission oriented layers
3 Network Layer	
2 Data Link Layer	
1 Physical Layer	

Fig. 2.2-1: The seven layers of the OSI reference model for communication.

The functions in each layer are as follows ([KDS+00]):

1. Physical Layer: Layer 1 provides physical support to transfer the data between the end points of a link. It specifies electrical, mechanical, procedural, and functional rules of exchange. This layer is the only one in the OSI model that physically interfaces with a physical layer of another end point. There are different ways and topologies to connect network nodes together: a bus system, through a centralized switching unit, a ring topology, etc.

2. Data Link Layer: Layer 2 permits the error free exchange of data on a communication link. It may also provide link level flow control and synchronisation. Also, in some cases the access to the network is controlled in this layer. Communication interfaces are identified by a so called MAC address (Media Access Control Address). Example: This layer defines the framing, addressing and checksumming of Ethernet packets.

3. Network Layer: Layer 3 provides services such as routing, network level flow, and congestion control. It defines the protocols capable of routing the data through one or more intermediate communication nodes.

4. Transport Layer: Layer 4 guarantees a constant quality of service for data transfer to the higher layers regardless of the type of network actually used. QoS may be defined in terms of delay, loss rate, and priority assignment.

5. Session Layer: Layer 5 defines the organization of the dialogue between distant applications. It is responsible for session establishment between distant applications. Furthermore, it provides support during connection recovery in case of failure and disruption of communication between processes at the end points. At this layer usually the login names of users at the end points of communication are accessible.

6. Presentation Layer: Layer 6 permits systems which exchange data to interpret these independent of their syntactical representation in the system. Presentation layers of end points may exchange control information in order to negotiate a common data format syntax.

7. Application Layer: Layer 7 provides an interface to the user, e.g. an application program such as e-mail or file transfer.

In the OSI model, communication between corresponding layers and adjacent layers is subject to strict rules. A lower layer `(N-1)` is service provider to its immediate upper layer `(N)`. An upper layer is user of services from the lower layer. Neighbouring layers are isolated from each other and communicate only by using so called primitives. A layered communication model is also called a **protocol stack**.

protocol stack

In each layer (except perhaps at the application layer), a packet has two parts: the header and the body. The header contains protocol information relevant to that layer, while the body contains the data for that layer, which often consists of a whole packet from the next layer in the protocol stack. Each layer treats the information it gets from the layer above it as data, and applies its own header to this data. At each layer, the packet contains all of the information passed from the higher layer. This process of preserving the data while attaching a new header is known as **encapsulation**.

encapsulation

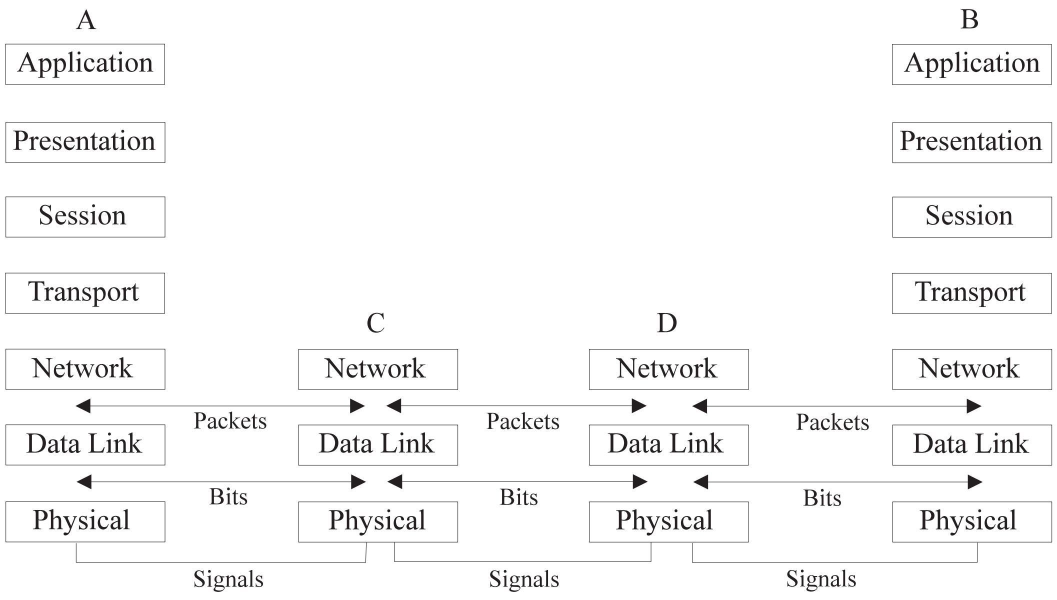


Fig. 2.2-2: An exemplary network

In Fig. 2.2-2 a typical network model between two computers (or users) `A` and `B` linked across a network is shown. The two computers are not connected directly physically, there are intermediate network units `C` and `D`. In the network (intermediate) nodes, like `C` and `D` in the figure, only the layers 1 to 3 are usually implemented.

In the following paragraphs we will describe the main principles of **switching techniques** in communication networks. Basically, we differentiate two switching principles between two communicating processes `A` and `B` through a network, namely circuit switched and packet-switched communication.

switching techniques

circuit switching

1. **Circuit Switching**

   In circuit switching, the communication partners A and B exclusively use a communication link or channel across the network during the whole communication process. Nobody else can use this link or channel at the same time. The switching process can be subdivided into three phases:

   a. Setup Phase: A indicates to the relevant switching unit that he wants to communicate with B. The switching unit informs B about A's request. If B agrees, the link between A and B is exclusively reserved.

   b. Connection Phase: A and B exchange information over the channel.

   c. Termination Phase: Both communication partners can inform the switching unit that the communication is completed.

packet switching

2. **Packet Switching**

   In packet switching, the message which is to be transmitted from A to B is divided into packets. Each packet is provided with destination and control information and is separately transmitted via the network. Packet switching can be realized in two different operational modes: connectionless and connection-oriented:

   a. Connectionless mode (or datagram mode): Each packet of the communication process is provided with destination and control information as well as a sequence number. In this case no setup and termination phase is required. The packets are sent to the destination independent from each other. Thus, packets can take different paths across the network and possibly overtake each other. By using a sequence number for each of the packets, the receiver can reorder the packets and reassemble them to the original message. The datagram packet mode is also used in the Internet protocol IP and UDP. A special case of connectionless packet switching is the message switching. In this case the whole message is sent within one packet from the sender to the receiver. In contrast to circuit switching, no direct path of transmission exists between the participants but the message is stored temporarily in intermediate nodes. The message is provided with address and control information, temporarily stored in intermediate switching units, and after passing several switching units transferred to the receiver.

   b. Connection-oriented mode: Before starting transmission in the connection-oriented transmission mode, the path from participant A to B across the network is determined. Thus a virtual circuit is set up. Similar to circuit switching we can distinguish three different phases of a virtual circuit: The setup, the connection and the termination phase. Each transmitted packet takes the same path via the network. In this way, the packets arrive at their destination in correct order and no additional sequence number is required. Packets which belong to a virtual circuit only need reduced address information to reach their destination. Consequently, the packet overhead is reduced. During the connection phase of the virtual circuit, the links contained in its transmission path are not exclusively available for it alone but may also be used by other virtual circuits in the network as well. An example for an connection-oriented protocol is the Internet protocol TCP.

Computer networks can be divided into several categories depending on their size:

1. LAN (Local Area Network): In a LAN personal computers, workstations, printers and servers are connected locally. A LAN can extend up to about 10km. Examples of LAN technologies are: Ethernet, FDDI, etc.

2. MAN (Metropolitan Area Network): A MAN extends across a city or a district within a city, across the area of a bigger enterprise or a university. A WAN can extend up to about 100km.

3. WAN (Wide Area Network): AWAN connects computers or smaller networks within one or several countries.

4. GAN (Global Area Network): A GAN connects computers distributed all over the world. It is realized by attaching different LANs and MANs with public or private long distance links.

Due to technical reasons, it is not always possible to attach each node to one particular network. Furthermore, it may be preferable due to performance and security reasons to include certain nodes in different networks. This may be to avoid network congestion or to restrict access to ensure data security. Apart from these aspects it may be desired to link comparable or technically different networks together. Therefore special network nodes exist, called **gateways** in general. Examples are hubs, bridges, or routers. A gateway connects technically different networks leading to a heterogeneous network. Two similar networks can be attached via a bridge. A bridge supports less functionality than a router which connects different types of networks. Gateways can operate on different layers in the protocol stack.

gateway

The binding elements between different networks are **routers**. A router is a special kind of network node which is connected to multiple physical networks via interfaces. For example, in packet-switched networks the traffic on the networks consists of packets (datagrams) which originate and end in the terminal equipment of the users hosts and are forwarded by the routers according to the router tables. These routing tables are stored in the routers and specify the path, the next interface of the router or the next node the datagrams will take through the network. These routing tables can be updated statically or dynamically. In a router the headers in the packets could be modified.

routers

An important communication principle is the **client-server architecture**. The operation modes of central computers started with batch processing. They further developed to time-sharing processes. In this case, several computers, called terminals, are connected to the central computer via data communication links. The jobs of

client-server architecture

the terminals are processed piecewise in parallel by the central computer. Since the 1970'ies microprocessors with increasing speed and decreasing size are coming up constantly. This has resulted in a growing use of microcomputers allowing the user to work independently. But the need to connect computers has not decreased. Networking computers allow to share peripheral equipment such as printers and perform distributed computing. A distributed system is usually realised in form of a client- server architecture. The server is a fast computer with high amount of storage capacity offering its service to clients. At the server a server program for that service is started. The server program waits for requests from the clients on a special network port. If a request message arrives at the server, it processes the request, and sends a response message to the client. Connection-less and connection-oriented client-server services are possible.

ISO 7498-2, X.800

In **ISO Standard 7498-2** [ISO89] (adopted as **recommendation X.800 by ITU-T**) general elements of a security architecture for communication protocols based on the OSI reference model are defined. The objective of OSI is to permit the interconnection of heterogeneous computer systems so that useful communication between application processes may be achieved. At various times, security controls must be executed in order to protect the information exchanged between the application processes. Such controls should make the cost of improperly obtaining or modifying data greater than the potential value of doing so, or to make the time required to obtain the data improperly so large that the value of the data is lost. X.800 defines the general security-related architectural elements which can be applied appropriately in the circumstances for which protection of communication between open systems is required. It establishes, within the framework of the reference model, guidelines and constraints to improve existing recommendations or to develop new recommendations in the context of the basic OSI reference model in order to allow secure communications and thus provide a consistent approach to security in OSI. X.800 extends the basic reference model to cover security aspects which are general architectural elements of communications protocols, but which are not discussed in the basic reference model. Scope and field of application of X.800 are:

1. Providing a general description of security services and related mechanisms, which may be provided by the reference model.
2. Definition of positions within the reference model where the services and mechanisms may be provided.

Basic security services and mechanisms and their appropriate placement have been identified for all layers of the basic reference model. In addition, the architectural relationships of the security services and mechanisms to the reference model have been identified. Additional security measures may be needed in end systems, installations and organizations. These measures apply in various application contexts. The definition of security services needed to support such additional security measures is outside the scope of X.800. X.800 is a very abstract and general definition of security services.

2.3 The Internet protocol family

In this section we describe the basic facts about the protocols used in the Internet, namely the IP protocol family. This protocol family is used to transmit and provide the well known services and protocols on the Internet like FTP, SMTP, HTTP, ... An overview of these services is given in Section 2.3.8. The Internet protocol family mainly consists of the following protocols: IP (Internet Protocol), ICMP (Internet Control Message Protocol), UDP (User Datagram Protocol), and TCP (Transport Control Protocol). We will describe these protocols and their message formats in more detail in the following sections. There are a lot of security problems in the IP protocol family and the services, which are based on these protocols (Section 2.3.7).

2.3.1 The history of the Internet

The basic task of the Internet is to globally transport data from one location to another, independently of the network protocols used at the OSI layer 1 and 2. This allows communication among users, exchange of information as well as collaborative use of software and computers.

Arpanet

Todays Internet can be traced back to the **Arpanet**. The Arpanet was funded by the Advanced Research Projects Agency (ARPA) in the U.S. Department of Defense (DoD). The development of the Arpanet began in 1966 as an experiment to test the new packet switching technology and protocols that could be used for distributed computing. In 1969, the Arpanet consisted of four packet switched nodes, connecting a few computers and terminals. Until 1972 when the first package for electronic mail was written, the two main applications of the Arpanet where remote computing and file transfer. Electronic mail became of growing importance so that only one year later three quarters of the Arpanet traffic arouse from electronic mails.

The packet switching technology of the Arpanet was so successful that ARPA also applied it to radio communication and satellite communication. But due to the different environments of the three networks, certain parameters such as the packet size were different. To be able to integrate these three networks, in 1974, a first approach to the transmission control protocol was published. This approach was split in 1978 and led to the protocols providing the foundation of the Internet. The Arpanet became just one of the connected networks. In the years 1982-1983, the Arpanet converted from its original network (NCP) to the IP protocol family. In 1983, the name server was developed at the University of Wisconsin. Since then, it was no longer necessary to know the IP network number of the destination host a packet should be sent to. This led to the introduction of the Domain Name System (DNS) about one year later. In the following years the Arpanet was extended to include computer science research groups and companies. In 1990, the original Arpanet was finally shut down and replaced by the so called Internet. At CERN (European Laboratory for Particle Physics), a distributed hypermedia technology to facilitate the international exchange of research information using the Internet was proposed in 1989. Two years later, a prototype of the World Wide Web in form of a WWW server and browser supporting HTTP (Hypertext Transfer Protocol) was developed at CERN.

IAB, IETF In 1986 the **IAB (Internet Architecture Board)** established the **IETF (Internet Engineering Task Force)**. The IETF is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. It is open to any interested individual. The actual technical work of the IETF is done in its working groups, which are organized by topic into several areas (e.g., routing, transport, security, etc.). Much of the work is handled via mailing lists. The IETF holds meetings three times per year. The IETF working groups develope and specify new protocols or publish informal documents. At the beginning of developing a new protocol for the IP protocol family an Internet Draft has to be published by the IETF. After discussion and further developing of the new protocol the revised document can get the status of an RFC (Request for Comments) ([Cha92]). Fundamental RFCs, like the specification of the IP, TCP or UDP protocols get the status of an Internet Standard. The Internet Drafts and RFCs are published at http://www.ietf.org.

2.3.2 Internet protocol stack

There is no official Internet network model as there is in case of OSI. But based on the protocol standards that have been developed, the communication tasks of the several Internet protocols can be organized into four relatively independent layers (see Table 2.3-1). This model is also called the TCP/IP protocol stack.

Tab. 2.3-1: *The TCP/IP protocol stack.*

Layer	Examples of protocols
Application Layer	FTP, Telnet, HTTP, SMTP, . . .
Transport Layer	TCP, UDP, ICMP, . . .
Internet Layer	IP
Network Access Layer	Ethernet, FDDI, ATM, GSM, ISDN, . . .

At the application layer, the packets simply consist of the data to be transferred. For example, a part of a file being transferred during an FTP session or a keystroke from the user on the terminal during a Telnet session. As this packet moves to the transport layer, the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP) preserves the data from the previous layer and attaches a header to it. At the next lower layer, the Internet layer, the Internet Protocol (IP) considers the entire packet, which is now consisting of the TCP or UDP header and the data from the application layer, to be data and attaches its own IP header. Finally, at the network layer, Ethernet or another network protocol, it considers the entire IP packet passed to it to be data and attaches its own header (see Fig. 2.3-1). At the other side of the communication, this process is reversed. As the data is passed up from one layer to the next layer, each header is stripped off by its respective layer.

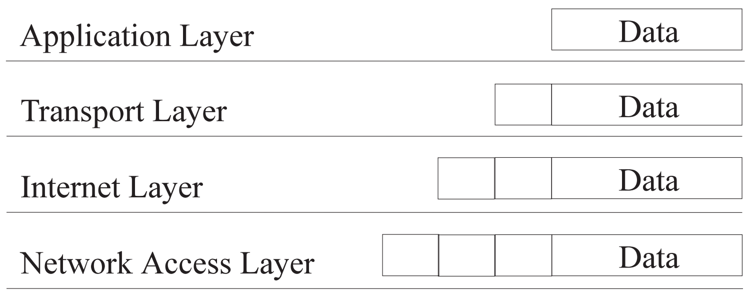


Fig. 2.3-1: The building of packets in the Internet protocol model.

The tasks in each layer of the TCP/IP protocol stack are as follows:

1. Network Access Layer

   This layer corresponds to the physical and data link layer of the OSI model. This network access layer is concerned with the exchange of data between an end system and the network to which it is attached. It is responsible for the access to and transmission of data across a network for two end systems which are attached to the same physical network. Also the tasks of the data link layer of the OSI model has to be realized in the network access layer. At the data link layer, data is usually organized into units called frames. Each frame has a header that includes address and control information and a trailer that is used for error detection or correction.

2. IP Layer (or Network layer)

   This layer corresponds to the Network Layer of the OSI model. At the IP layer, data is routed across gateways or routers. The IP protocol operates at this layer to transport and route packets across networks independent of the network medium. Data may traverse a single link or may be relayed across several links through intermediate network nodes, like routers and gateways. The packet format of IP packets is outlined in Fig. 2.3-3. The IP layer operates connectionless because every IP packet is transported and routed independently and IP does not guarantee reliable or in-sequence of datagrams.

3. Transport Layer

   This layer corresponds to the transport layer in the OSI model. The transport layer manages the flow of data between two hosts. It relies on two transport protocols, TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). TCP provides reliable data communication to applications as a connection oriented service. It contains mechanisms which guarantee that data is delivered error-free, without omissions and in-sequence. UDP is classified as a connectionless protocol. It is much simpler as TCP and does not than offer reliability guarantees, flow control, nor error-recovery measures. It is sometimes used in place of TCP in situations where the full service of TCP is not needed.

4. Application Layer

This layer includes the session, presentation and application layer from the OSI model. Data is received as commands from the user and as data from the network application on the other end of the connection. Operations such as electronic mail and file transfer are provided in this layer.

2.3.3 IP

We now want to take a closer look at the IP addressing, the IP protocol and the packet format in the IP layer.

In the Internet, IP addresses are used to identify hosts and route data to them. Hence, every network node in an IP Network must have a valid IP address and should have a valid name. If a network node has more than one network interface, it could happen that this node has an IP address for each interface. Above the IP layer, in the network access layer there exist different addresses which are called MAC addresses. The host name and the IP address must be unique in the network. A host name is translated into its IP address by looking up the name in a local file or database or with the DNS (Domain Name Service) service.

IP addresses consists of 32 bits, usually represented in a dotted decimal format. This means that the addresses are expressed with decimal digits, each separated by a dot. An IP address may look like `a.b.c.d`, where `a`, `b`, `c` and `d` are numbers between zero and 255. The IP address could be divided in a network and a host part by a

subnet mask

subnet mask. A subnet mask is a 32 bit number which is also expressed in a dotted decimal format. With the subnet mask, a host can decide locally if the the destination host address is on the same local network (direct routing) and physically connected to that host, or if the destination host is on another network (indirect routing). In the latter case the IP packet has to be sent first to the MAC address of the next router or gateway, which routes the packet to the right destination network. The source IP number and the destination IP number are each bitwise AND operated with the network mask. If the two numbers are equal then the destination host is on the same local network. In the other case, the destination host in on another network.

Example 2.3-1: Subnet mask
The IP address of a host is `123.76.12.211` and the local subnet mask is `255.255.255.0`. The AND operation with source IP address and subnet mask results in `123.76.12.0`. See also Fig. 2.3-2. We now consider the following two cases:

1. The destination IP address is `123.76.12.89`. The AND operation with destination IP address and subnet results in `123.76.12.0`. So the destination host is on the same local subnetwork.

2. The destination IP address is `123.76.15.70`. The AND operation with destination IP address and subnet mask results in `123.76.15.0`. So the destination host is on another local subnetwork.

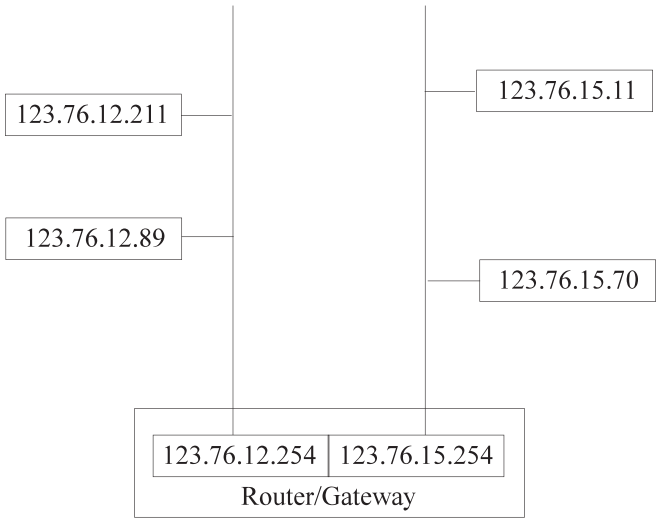


Fig. 2.3-2: Network for Example 2.3-1. There are two local networks, which are connected through a router. The router has two network interfaces, each connected to one local network and with IP address `123.76.12.254` and `123.76.15.254`. On each local network there are two hosts.

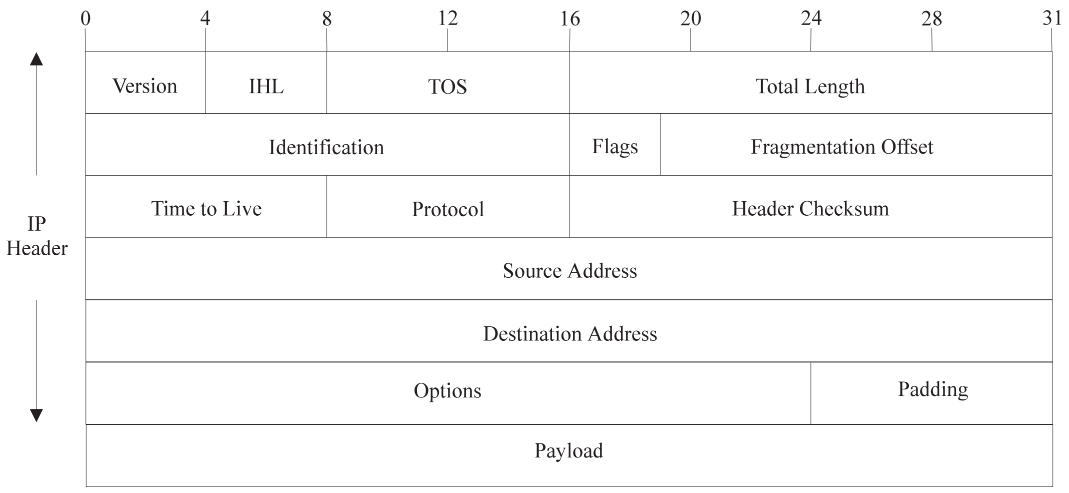


Fig. 2.3-3: The IP packet format. The numbers are bit positions. A word is formed by 32 bits.

In the IP layer, the IP packet is made up of two parts: the IP header and the IP body (payload), as shown in Fig. 2.3-3. Our description follows the current version of IP, namely Version 4 or **IPv4** ([Pos81b]). The latest version **IPv6** has already been standardized and deployed experimentally ([DH98]). The**IP packet** contains the following fields.

IPv4, IPv6

IP packet format

1. Version (4 bit): This field indicates the version of the used IP protocol. IPv4 is currently used.

2. IHL (Internet header length, 4 bit): Consists of the length of the header of the IP datagram in words (32 bits). The minimum length of the IP header is 20 byte.

3. TOS (type of service, 8 bit): This field consists of eight bits, but only four bits are actually used to make type of service requests to routers. Often this field is unused and holds the value zero.

4. Total Length (16 bit): The value stored in this field represents the entire datagram length, including the header (in bytes).

5. Identification (16 bit): The originating host specifies a unique 16- bit identifier for every datagram. This identification is required as the datagrams may be fragmented on their way through an internetwork. Fragmentation of IP datagrams can occur when they cross network boundaries between networks which may have different maximum transfer units (MTUs) at the network access layer. Notice also the following two items.

6. Flags (5 bit): The first of these bits is unused, the other two are used to control the way a datagram is fragmented. If the DF (don’t fragment) bit is set to 1, the datagram must not be fragmented. If the datagram has to be fragmented, the router throws it away and sends an error message to the sending host. When the MF (more fragments) bit is set to 1, then the datagram is one of several fragments, but not the last one. If the MF bit is set to 0 this indicates that this datagram is the last of the number of datagrams or that the datagram has not been fragmented.

7. Fragment Offset (13 bit): This number informs the receiver how many units (one unit equals eight byte) the current datagram is apart from the start of the original.

8. Time to Live (TTL, 8bit): This field indicates how long a datagram is allowed to exist after entering the internetwork. The maximum TTL is 255. As a datagram is forwarded by a router, its TTL is decremented by one. If the TTL has reached zero, the datagram is discarded and the router sends an error message to the sending host.

9. Protocol (8 bit): In this field, the protocol number of the next higher layer (ICMP (=1), TCP (=6), UDP (=17), . . .) is identified. This protocol numbers are defined in [RP94].

10. Header Checksum (16 bit): This field contains a checksum of the IP header. By computing a checksum, the intermediate routers and the destination host can detect transmission errors in the IP header.

11. Source Address (32 bit)

12. Destination Address (32 bit): Special destination addresses are: `127.0.0.1` (the local host), `255.255.255.255` (all hosts on the local subnetwork).

13. Options (variable bit length): Up to 40 extra IP header bytes are available to carry one or more options. The options that are included in a datagram are chosen by the sending application. Examples are:

a. Reverse Route: The traffic flowing back from destination to the source must follow the same path.

b. Record Route: A header field contains a list of IP addresses of routers visited by the IP datagram.

c. Source Route: This option lets the source of a packet specify the route the packet is supposed to take to its destination, rather than letting each router along the way use its routing tables to decide where to send the packet next. Source routing is supposed to override the instructions in the routing tables. In theory, the source routing option is useful for working around routers with broken or incorrect tables. If you know the route that the packet should take, but the routing tables are broken, you can override the bad information in the routing tables by specifying appropriate IP source route options on the IP packets. In practice though, source routing is commonly used only by attackers who are attempting to circumvent security measures by causing packets to follow unexpected paths. So most routers are configured to ignore this option.

d. Timestamps: If the timestamp option is set, then every router on the packets path from source to destination hosts attaches a current time mark on the packet.

e. There exist several security options, like IPsec. We will discuss these in chapter 3.

14. Padding (maximal 31 bit): Padding is used to make the header length a multiple of 4 bytes if the options do not end on a 4-byte boundary.

15. Payload (variable bit length): The data from higher layer protocols.

2.3.4 ICMP

The IP protocol aims at one major task: to move data from its source to its destination. But network layer protocols have to perform more tasks than transfering data. Systems rely on the network layer to coordinate different aspects of their operations including discovery of neighbours, controlling address assignments, and managing group membership. Furthermore, these protocols assist in reporting errors and providing diagnostic support.

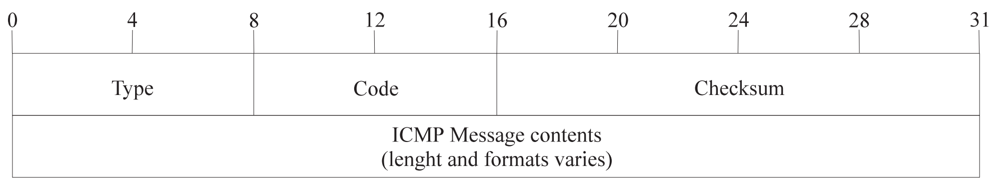


Fig. 2.3-4: The ICMP packet format. The numbers are bit positions.

In the Internet protocol family, these functions are assigned to the Internet Control Message Protocol (ICMP). ICMP is used for IP status and control messages. ICMP packets are carried in the payload of IP packets, just as TCP or UDP packets are.

ICMP message format

The **ICMP message format** is shown in Fig. 2.3-4 ([Pos81a]). It consists of the following fields:

1. Type (8 bit): This field indicates the function the message fulfills.
2. Code (8 bit): It includes further information about the content of the message, e.g. more specific description of errors.
3. Checksum (16 bit): The checksum is applied to ICMP message.
4. Payload (variable length and contents)

ICMP messages

Examples of **ICMP messages** include:

1. Echo request: What a host sends when you start the ping program/ service.
2. Echo response: What a host responds to an echo request with.
3. Timestamp: Request and reply messages that probe the round-trip time and find out the clock setting of the target system.
4. Address Mask: Request and reply messages that allow systems to discover the address mask which should be assigned to a network interface.
5. Routing information: ICMP messages can be applied to exchange routing information. By periodically broadcasting the current routing preferences, routers ensure that the hosts in their networks do not try to use a router that is inappropriate.
6. Time exceeded: This type of message is send, when a router determines that a packet appears to be looping.
7. Destination unreachable: This message returns a router, when the destination host of packet can’t be reached for some reason, e.g. because a network link is down.
8. Redirect: What a router sends to a host in response to a packet that the host should have sent to a different router. The router handles the original packet anyway, and the redirect tells the host about a more efficient path for the next time.

2.3.5 UDP

UDP (User Datagram Protocol) is a simple protocol in the transport layer of the Internet protocol stack. It provides no error detection, no error correction, no connection-oriented links, no handshaking, and no verification for delivery order. This tasks is defered to higher layers. UDP only offers the basic datagram delivery.

Applications and services based on UDP are often simple so that they do not need to maintain connections. UDP applications may consist entirely of requests and replies to requests. UDP provides a connectionless delivery service between two hosts, offering a service over a particular protocol port. Port numbers refer to processes running on network systems. Protocols at the transport layer no longer refer to specific nodes or network interfaces. Transport layer protocols identify source and destination processes in form of **port numbers**.

port numbers

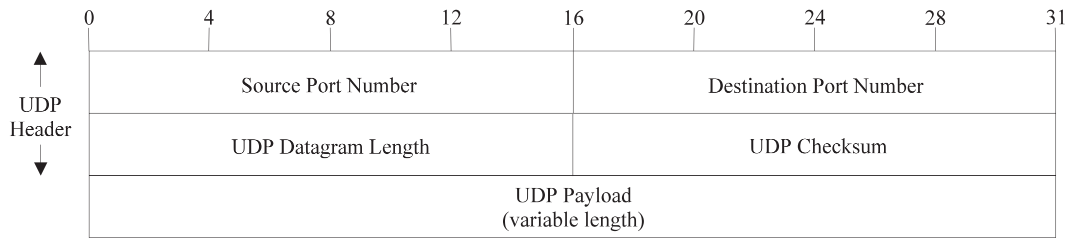


Fig. 2.3-5: The UDP packet format. The numbers are bit positions.

UDP is easy to implement and requires minimal overhead. Each UDP datagram contains a single message which may be a request or a reply. A UDP packet is transmitted as payload of an IP packet according to packet-switching in datagram modes. This means that it travels independent from other UDP packets across the network. The **UDP packet** structure is depicated in Fig. 2.3-5 ([Pos80]). It consists of the following fields:

UDP packet format

1. Source port (16 bit): The source port number of a datagram is determined by the host which generates it.
2. Destination port (16 bit): The destination port specifies a particular program or process running on the destination host. The sending host would have to query the destination host and ask for the right port number before it could send its datagrams.
3. UDP datagram length (16 bit): This field describes the length of the UDP packet in bytes.
4. UDP Checksum (16 bit): A checksum over the whole UDP packet, including UDP header and UDP payload. The checksum is calculated by using a pseudo-header that uses some information from the IP packet header.
5. UDP payload (variable length).

2.3.6 TCP

TCP (Transmission Control Protocol) is a protocol in the transport layer in the IP protocol stack. In contrast to UDP, which offers little support concerning reliability or guarantees, TCP reliably connects hosts across a network. TCP operates connection oriented. Each TCP connection behaves as if a direct two-way connection (bidirectional) between the communicating hosts exists. The communication consists of three phases: The setup, the connection and termination phase. TCP provides endtoend reliability, requiring that communicating hosts coordinate and agree to make connections and acknowledge receipt of network traffic. Each UDP datagram is an individual message or reply. In comparsion to that, each TCP segment is related to the segment before and after it. The first and the last segments in a sequence require special treatment.

TCP socket

Each TCP connection is identified uniquely with a combination of each host's IP address and port number for the connection. The combination of port number and IP address of the host on which the process is running is referred to as a **TCP socket**.

Example 2.3-2: Telnet server and client processes
Consider two hosts with IP addresses `128.36.1.24` (client) and `130.42.88.22` (server). A Telnet server process is running on the server on port `23`. Hence the TCP socket of the Telnet server process is (`130.42.88.22,23`). If the client wants to connect with the Telnet server, it opens a TCP socket (`128.36.1.24, X`), where `X` is a not used port number on the client, for example `X = 2258`.

TCP reliability

Concerning reliability, only UDP offers the checksum which is calculated from the pseudo-header and the UDP datagram. It helps the receiving process to verify that the datagram arrived at the right address without corruption. TCP makes three guarantees concerning **reliability** to the application layer:

1. The destination will receive the application data in the order it was sent.
2. The destination will receive all the application data.
3. The destination will not receive duplicates of any of the application data.

TCP will kill a connection rather than violate one of these guarantees.

Example 2.3-3: Reliability of TCP in the case of packet loss
If TCP packets from the middle of a connection session are lost in transit to the destination, the TCP layer will arrange for those packets to be retransmitted before handing the data up to the application layer. It will not hand up the data following the missing data until it has the missing data. If some of the data cannot be recovered, despite repeated attempts, the TCP layer will kill the connection and report this to the application layer, rather than hand up the data to the application layer with a gap in it.

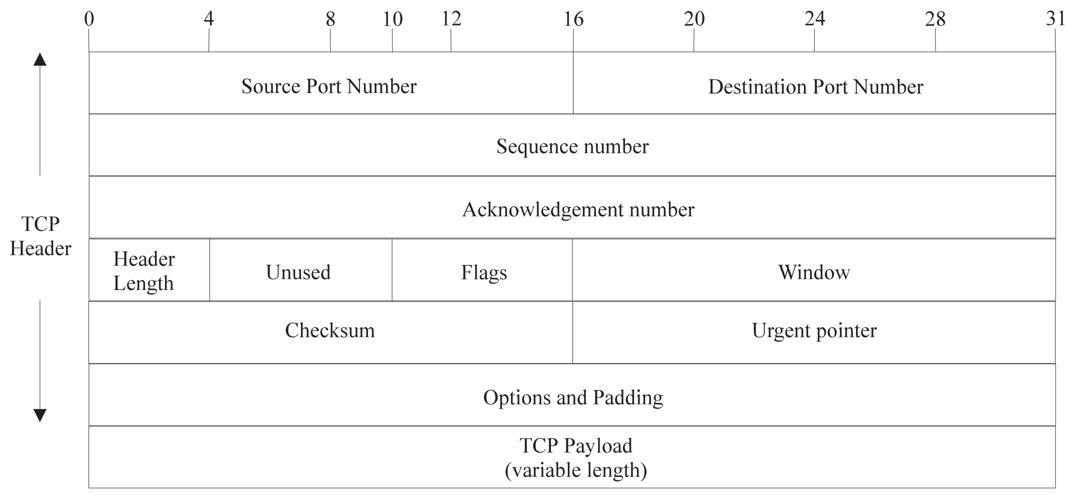


Fig. 2.3-6: The TCP packet format. The numbers are bit positions.

The TCP packet structure (see Fig. 2.3-6) and TCP protocol is defined in [Pos81c]. The fields of the **TCP packet** have the following meaning:

TCP packet structure

1. Source port number (16 bit) and destination port number (16 bit): The initiating (source) host assigns itself an ephemeral port number (from `1024` to `5000`), which is usually a randomly chosen value greater than `1023`. The destination port number is the well-known port associated with the service requested from the remote host ([RP94]).

2. Sequence number (32 bit): Each byte of a TCP byte stream in a connection is numbered, starting with an arbitrary number selected by the sending host. TCP connections are duplex (bidirectional) which means that data is transmitted in both directions at the same time. Each host selects an arbitrary starting point for numbering the bytes of its own data stream. The sequence number in the TCP header indicates the number the sending host has assigned to the first byte in the current segment. The numbering starts at an arbitrary number between 0 and $2^{32} - 1$ and restarts at zero when the highest vale has been reached.

3. Acknowledgement number (32 bit): This field contains the value of the sequence number which is expected next from the other side. It serves also as acknowledgement for the received bytes with a value minus one of this field. If the acknowledgement does not arrive within a timeout interval, the data is retransmitted.

4. Header length (4 bit): This field indicates the length of the TCP header in words (4 bytes). The length of the TCP header is not fixed as it may be extended when using TCP options. The length of the TCP header varies between 20 and 60 bytes.

5. Unused (6 bit):

6. TCP flags (6 bit): The TCP flag bits are employed to negotiate and manage connections.
    - URG (urgent): If set to 1, urgent data is included in the segment and the urgent pointer in the TCP header is used to point at the urgent data.
    - ACK (acknowledgement): Indicates that the acknowledgement number in the segment header is valid. If this flag is set to zero, the acknowledgement field must be ignored. The flag is zero as long as no segment from the other side of connection has been received. In this case there is nothing to acknowledge. Once a connection is established, it should always be set to 1.
    - PSH (push): The data should be delivered to the application as soon as possible.
    - RST (reset): This flag indicates an error, the connection must be reset or killed.
    - SYN (synchronize): During setup of a TCP connection, this bit is set to indicate that the synchronization of sequence and acknowledgement numbers takes place.
    - FIN (finish): The sender has no more data to send and wants to finish the connection.
7. Window size (16 bit): The window size is a dynamic value that varies depending on how much data the process at either end of the circuit is willing to accept at any time. The window size is specified as a number of bytes that the receiving host accepts within its window. The window size may depend on e.g. the processing speed or the temporal storage capacities of the receiving host. The process at either end can modify its window size to increase the efficiency of the link. Smaller windows mean that the receiving host cannot process the incoming data quickly enough to keep up with the current pace.
8. Checksum (16 bit): The checksum value is calculated with a TCP pseudoheader.
9. Urgent Pointer (16 bit): The urgent pointer is only employed if the URG flag bit is set. It is used to indicate the sequence number of the last byte which is part of the urgent data.
10. Options and Padding (variable bit length): The most common option is to specify a segment size. Hosts can avoid fragmentation of TCP fragments by letting the host on the other end know the largest segment size it is willing to accept. The padding mechanism ensures that the TCP header length is a multiple of 32 bits.
11. TCP payload (variable length).

To set up a TCP connection between two hosts, the so called **three-way handshake** has to be performed. The name of the procedure stems from the fact that three messages SYN (for synchronization), SYN, and ACK (acknowledgement) have to be exchanged to start the TCP connection. The setup between two hosts A and B consists of the following three steps:

three-way handshake

1. SYN ($A \rightarrow B$): Host A sends a TCP packet to B in order to request B to open a TCP connection to A. The initial sequence number which has been randomly chosen is transmitted to B. The SYN flag is set indicating that the circuit is being synchronized. The ACK flag is set to zero.

2. SYN ($B \rightarrow A$): Host B sends an acknowledgement of the initial packet to host A. The ACK flag is set to 1. The opening sequence number form A is incremented by one and written into the acknowledgement field. Host B chooses its random sequence number. The SYN flag is set to 1. When host A receives the packet, the connection from A to B is established, but the link from B to A has to be validated.

3. ACK ($A \rightarrow B$): Host A responds to host B's acknowledgement with an acknowledgement (ACK=1). The sequence number of B is incremented by one and put into the acknowledgement field. The SYN flag is set to zero because the synchronization is complete as A responds with this message. The responding message gets a by one incremented sequence number from host A.

When these three steps are completed, both sides can start sending data.

2.3.7 Security problems with the IP protocol family

There are several security risks in the IP protocol family and possible attacks. We will discuss a few of them here ([ZCC00]):

1. Implementation Weaknesses

   Many attacks at the Internet layer or transport layer are denial of service attacks that exploit weaknesses in implementations of the IP protocol family to crash hosts, e.g. send overlapping fragments. There are also attacks that send invalid combinations of options, set invalid length fields, or mark data as urgent when no application would.

2. IP Spoofing

   In IP spoofing, an attacker sends packets with an incorrect source address in the IP packet header. When this happens, replies will be sent to the apparent source address, not to the attacker. This might seem to be a problem, but actually there are three cases where the attacker doesn't care:

   a. The attacker can intercept the reply.

   b. The attacker doesn't need to see the reply.

   c. The attacker doesn't want the reply.

3. Packet Interception

   Reading IP packets (or upper layer packets) as they go by on the physical network, frequently called packet sniffing, is a frequent way of gathering information. If one is passing around important information unencrypted, and this is the normal operation mode of the IP protocol family, it may be all that an attacker needs to do.

   In order to read a packet, the attacker needs to get the packet somehow. The easiest way to do that is to control some machine that the traffic is supposed to go through anyway (a switch, a router, or a firewall). These machines are usually highly protected, however, and do not usually provide tools that an attacker might want to use.

   On some networks, like Ethernet or Token-ring, it is possible to get access to packets very easyly. An Ethernet network that uses bus topology, or that uses 10-base T cabling with unintelligent hubs, will send every packet on the network to every machine. Tokenring networks, including FDDI rings, will send most or all packets to all machines. Machines are supposed to ignore the packets that are not addressed to them, but anybody with full control over a machine can override this and read all the packets, no matter what destination they are sent to.

   An attacker can also intercept packets and change the contents of a packet and send it to the destination host. This is possible since no authentication and integrity check mechanisms exist in the normal implementations of the IP protocol family.

4. Attacks on fragmented IP packets

   Attackers can use specially fragmented IP packets to conceal data. Each IP fragment contains information where the data it contains starts and ends. Normally, each one starts after the last one ended. An attacker can construct IP packets where fragments actually overlap, and contain the same data addresses. This does not happen in normal operation. It can only happen when bugs or attackers are involved.

   Operating systems differ in their response to overlapping fragments. Because overlapping fragments are abnormal, many operating systems respond very badly to them and may reassemble them into invalid packets, with the expected sorts of unfortunate results up to and including operating system crashes. When they are reassembled, there are differences in whether the first or second fragment's data is kept. These differences can be increased by sending fragments out of order. Some machines prefer the first version received, others the most recent version received, others the numerically first, and still others the numerically last.

   There are three kinds of attacks possible by overlapping fragments:

   a. Simple denial of service attacks against hosts with poor responses to overlapping fragments.

   b. Information hiding attacks: If an attacker knows that virus detectors, intrusion detection systems, or other systems that pay attention to the content of packets are in use and can determine what assembly method the systems uses for overlapping fragments, the attacker can construct overlapping fragments that will obscure content from the watching systems.

   c. Attacks that get information to otherwise blocked ports: An attacker can construct a packet with acceptable headers (e.g. TCP or UDP headers) in the first fragment but then overlap the next fragment so that it also has headers in it. Since packet filters do not expect TCP headers in non-first fragments, they will not filter them, and the headers do not need to be acceptable.

5. Attacks on ICMP

   There have also been attacks that use ICMP, as a covert channel, a way of smuggling information. As we mentioned in Section 2.3.4, most ICMP packet bodies contain little or no meaningful information.

   However, they may contain padding, the content of which is undefined. For instance, if you use ICMP echo for timing or testing reasons, you will want to be able to vary the length of the packets and possibly the patterns of the data in them. Therefor it is allowed to put arbitrary data into the body of ICMP echo packets, and this data is normally ignored. It is not filtered, logged, or examined by anyone. For someone who wants to smuggle data through routers and firewalls that allow ICMP echo, these bodies are a very tempting place to put it in. They may even be able to smuggle data into a site that allows only outbound echo requests by sending echo responses even without having seen a request. This will only be useful if the target host that the responses are being sent to is configured to receive them. It will not help anyone break into a site, but it is a way for people to maintain connections to compromised sites.

6. Port Scanning

   Port scanning is the process of looking for open ports (TCP or UDP ports) on a machine, in order to figure out what services are running on the machine and what might be attackable. Straightforward port scanning is quite easy to detect, so attackers use a number of methods to disguise port scans. For instance, many machines are so configured, that they do not log connections until they are fully made. So an attacker can send an initial TCP packet, with a SYN but no ACK, get back the response, then another SYN if the port is open, and a RST if it is not, and then stop there. This is often called a `SYN scan` or a `half open scan`. Although this will not get logged, it may have other unfortunate effects, particularly if the scanner fails to send a RST when it stops.

Attackers may also send other packets, counting a port as closed if they get a RST and open if they get no response, or any other error. Almost any combination of flags other than SYN by itself can be used for this purpose, although the most common options are FIN by itself, all options on, and all options off. The last two possibilities, sometimes called `christmas tree` and `null`, tend to have unfortunate side effects on weak IP stack implementations. Many devices will either crash or disable the IP protocol stack.

For a further discussion of problems with the IP protocol family read the article [MF00].

2.3.8 Common Internet services and their security risks

There are a number of Internet services that users want and that most sites try to support. There are important reasons to use these services. But there are potential security problems with each of them. The Internet services are usually build over the ICMP, TCP or UDP protocols. This section briefly summarizes the major Internet services and provides a high level description of them without going into to much details. None of these services are really secure. Each one has its own security weaknesses, and each has been exploited in various ways by attackers. Before you decide to support a service at your site, you will have to assess how important it is to the legal users of this network and whether you will be able to protect them from its dangers. There are various ways of doing this:

1. Running the services only on certain protected machines,
2. using especially secure variants of the standard services, or
3. blocking the services completely to or from some or all outside systems.

More details on the protocols, message formats and security weaknesses of the above services will be discussed in the following chapters.

1. Naming services

   A naming service translates the names that people use and the numerical addresses that machines use. Different protocols use different naming services. The primary protocol used on the Internet is DNS (Domain Name System), which converts between host names and IP addresses. For the end user it is much easier to work with names than with numbers identifying networks and hosts. The resulting system of names is administered in a distributed database called Domain Name System (DNS) and used by the Internet software to resolve network and host names to find out their IP addresses. A **domain name** is a hierarchical name which is registered for an organisation. Each level gives more information about the respective domain. Examples for root domain names are `de` or `com`. Within each root domain (or top-level domain)

domain name

second level domains are assigned and so on. An example of a full domain name is `fernuni-hagen.de`.

In early days of the Internet, it was possible for every site or host to maintain a host table that listed the name and number for every machine on the Internet that it might ever care about. With millions of hosts attached, it is not practical for any single site to maintain a list of them. Instead, DNS allows each site to maintain the information about its own host.

The main risk in providing a DNS service is that you may give away more information than you intend. Also the DNS service is a distributed database consisting of many DNS servers on the Internet. This makes it easy for an attacker to install a deceitful DNS server distributing incorrect information. So, using DNS for authentication services make them vulnerable to such attacks. This can be handled by a combination of methods:

a. Using IP addresses, rather than host names, for authentication.

b. Authenticating users instead of hosts on the most secure services, because IP addresses can also be spoofed.

There are also other naming and directory services like WINS (Windows Internet Name Service) fromMicrosoft, LDAP (Lightweight Directory Access Protocol), NIS (Network Information Service) from SUN or NDS (Novell Directory Service) from Novell.

2. Electronic Mail

Electronic mail is one of the most popular network services. It is a service with relatively low risk, but this does not mean that it is risk-free. Forging electronic mail is trival, and forgeries facilitate two different types of attacks:

a. Attacks against your reputation and

b. social manipulation attacks.

The electronic mail system on the Internet consists of mail servers accepting, transporting and holding mail messages and mail clients to send and receive mail messages to/from mail hosts. **SMTP (Simple Mail Transfer Protocol)** is the Internet standard protocol for sending and receiving electronic mail. Mail messages are transported between mail servers and from a mail client to a mail server mainly via SMTP. Mail servers, like other programs have a tradeoff between features and security. The most common SMTP server in the Unix or Linux operating systems is Sendmail. Sendmail bas been exploited in a number of break-ins.

SMTP

While SMTP is used to exchange electronic mail messages between servers, users who read electronic mail that has already been deliverd to a mail server do not use SMTP. Across the Internet, the most common protocols for this purpose are **POP (Post Office Protocol) and IMAP (Internet Message Access Protocol)**. POP and IMAP both normally transfer user authentication

POP and IMAP

data (login name and password) and the email message without encrypting it, allowing attackers to read the mail and often to get reusable user credentials.

3. File Transfer and file sharing

   Electronic mail is only designed to transfer small files. Also you may want to have a single copy of a file but use it on multiple machines. This is called file sharing.

   FTP

   FTP (File Transfer Protocol) is the Internet Standard protocol for file transfer. With FTP it is possible to download (from the FTP server to your machine) or to upload (from your machine to the FTP server) files. So its is possible to bring in Trojan horse software and computer games in to your site by legitimate users.

   To get access to files made available by an FTP server, users log into the system using an FTP client with their system login name or a special login name (usually ”anonymous” or ”ftp”). In the later case most FTP servers request that users enter their own electronic mail address in response to the password prompt. In the authentification phase, the login name and password and in the file transfer phase, the content of the file are transmitted without encryption. There are alternatives to FTP using the SSH protocol (see chapter 3) to achieve file transfer in a secure manner.

4. Remote access

   Remote access can be used in situations in which one would like to run a program on a computer at another site. For instance, one has a slow computer with low storage facilities at the local site. The major questions concering security issues of remote access are:

   a. How are remote machines and users authenticated?

   b. What security mechanismes are used for the transmitted data?

   c. Can anyone take over a connection in progress?

   d. What commands can a remote user execute and what data can be accessed?

   Telnet

   Telnet is the standard for remote access on the Internet. Telnet allows remote text access for users from any Internet connected site without making special arrangements. Telnet sends all of its information unencrypted, which makes it extremely vulnerable to sniffing and hijacking attacks. For this reason, Telnet is now considered to be one of the most dangerous services when used to access a site from a remote systems. A mordern service which can replace Telnet and FTP is SSH (Secure Shell).

In chapter 4 we discuss the security of the services in the World Wide Web.

3 Internet Security Protocols

The Internet protocol suite (TCP/IP) has been designed with no thought of security. It is very easily possible to spoof IP addresses, to manipulate DNS, and to eavesdrop the links.

Many cryptographic protocols have been proposed, specified, and partly implemented for the Internet and the WWW in the recent past. The development of Internet standards for security (i.e. RFCs[5] published as a standard track documents) has been slow. The first RFC to introduce a security-oriented protocol into the TCP/IP suite did not appear until 1987, with the publication of the first e-mail security protocol specifications. In the following years, a number of security-oriented Internet protocols have been developed and are at various stages in the Internet standardisation process. Internet standards are formally developed under the auspices of the **Internet Society**, whose technical arm is the Internet Architecture Board (IAB). IAB consists of two taskforces: IRTF (Internet Research Task Force) and IETF (Internet Engineering Task Force). IETF consist of a large number of working groups (WG), where the bulk of standards developments takes place. Despite the greatly increased activity in security standards in the IETF, the Internet still lacks on security (the protocols developed must be used!).

Internet Society, IAB, IETF, IRTF

In this chapter we overview and briefly discuss some of the Internet security protocols. In the case of TCP/IP based networks, cryptographic security protocols can operate at any layer of the corresponding communications protocol suite. There are many proposals for providing security services at the network access, IP, transport and application layer. Here we focus on the IPSec set of security protocols in the IP layer and two transport layer security protocols, namely the secure socets layer (SSL) and the transport layer security (TLS) protocols.

3.1 Security at the Network Access Layer

As mentioned in chapter 2, in the TCP/IP communications protocol suite, the network access layer handles issues related to local area networking and is responsible for packet transmission on the physical media. Protocols that operate on this layer include Ethernet (IEEE 802.3), token bus (IEEE 802.4), token ring (IEEE 802.5), FDDI (fiber distributed data interchange), as well as protocols for dial-up networking, such as the serial line IP (SLIP), and the Point-to-Point protocol (PPP). For all practical purposes, Ethernet is most widely deployed for local area networking (LAN) and PPP is most often used for dial-up networking. Thus, security standards below the IP layer tend to address requirements associated with protocolls used to support IP transport - for example, link-by-link authentication.

Layer 1 in TCP/IP layer model

5 Request for Comment

Activities for LAN and MAN security

In the late 1980's, the IEEE started to address issues related to local area (LAN) and metropolitan area network (MAN) security: The IEEE 802.10 working group was established in 1988 for this purpose. Meanwhile, IEEE 802.10 specified several **standards for interoperable LAN/MAN security (SILS)** that are compatible with existing IEEE 802 and OSI standard specifications. Unfortunately, SILS has not been particularly successful, and there are hardly any products that implement the SILS specifications. Consequently, we do not address the work of IEEE 802.10 and SILS.

Link encryption

At this place we should mention the simple "**link encryption**": If there is a dedicated link between two hosts (i.e.the two entities involved in the communication have to be physically connected) and the traffic needs to be encrypted, one can use hardvare devices or software for encryption. The link encryption is the classic method of applying cryptography to digital communications. This is also the oldest approach to data security, and has been used for decades to protect the most sensitive military traffic and to provide secure links between banking organisations. An example is an authomatic teller machines (ATM) where all the machines are connected via dedicated links to a central bank office. If ATM machines were connected to an IP network instead of dedicated secure links, the data link layer security would not suffice and one would have to move up one layer to provide security services.

PPP authentication

Another example for security mechanism in this layer is the **PPP authentication**. The Point-to-Point protocol (PPP) was developed as a common interface for communication over point-to-point links (versus the transmission of IP over packet-switched networks). PPP is often used on dial-up links, enabling users to gain intermittent access to an Internet service provider via low-speed lines, and is also used in interrouter links. Authentication of the dial-up user or of the neighbour router is an important security concern, and is the focus of the security extensions to PPP (described in RFC 1334). PPP allows negotiation of the authentication technique, with the default being no authentication. Two forms of authentication are supported currently: simple passwords and challenge-response schemes. The protocol can be used to provide two-way authentication, so that each party can verify the identity of the other. Implementations of authenticated PPP used for interrouter communication typically perform two-way authentication, but dial-up users access via PPP typically involves only one-way authentication. The PPP authentication has been in many respects one of the successes among Internet security standards to date, although for other security features that may be applicable in PPP (e.g. confidentiality of traffic or continous authentiction and integrity of the traffic) no provision is made.

3.2 Security at the IP Layer

Layer 2 in TCP/IP layer model

The IP layer provides conectionless services and is responsible for routing of packets. Two important components in this layer are adressing and fragmentation (i.e. a packet is split into multiple IP packets at the source or in a router). After fragmentation, each packet is transmitted independently and is reassembled at the destination. The language of this layer is IP, the Internet Protocol. It provides the functionality for interconnecting end systems across multiple networks. For this purpose IP is implemented in each end system and in routers (devices that provide connection between networks). Higher-level data at a source end system are encapsulated in an IP protocol data unit (PDU) for transmission. The PDU is then passed through one or more networks and connecting routers to reach the destination end system.

IP and Security

One thing IP does not provide, though, is security. When the protocols of the TCP/IP family were designed, security was not considered at all. IP packets can be forged, modified, and inspected on their route. In 1994, the Internet Architecture Board (IAB) issued a report with the title ”Security in the Internet architecture” (RFC 1636), in which the general consensus was stated, that the Internet needs more and better security. The IAB included authentication and encryption as necessary security features in the next-generation IP (IPv6). This initiated the developement and standardisation of IPSec. Fortunately, these security capabilities were designed to be usable both with the current IPv4 and the future IPv6. Many vendors do now have some IPSec capability in their products.

IPSec developing

IPSec is a **set of general purpose protocols for protecting TCP/IP** communications and provide data source authentication, data integrity, confidentiality and protection against replay attacks. Actually, IPSec is an extension to the existing IP networking protocol. The authentication mechanism of IPSec ensures that a received packet was, in fact, transmitted by the party identified as the source in the packet header. In addition, this mechanism assures that the packet was not altered in transit. The confidentiality facility enables communicating nodes to encrypt messages to prevent eavesdropping by third parties. The key management facility is concerned with the secure exchange of keys. By implementing security at the IP level, an organisation can ensure secure networking not only for applications that have security mechanisms (E-mail, Web access), but also for many other security ignorant applications. In this section we give an overview of IP security (IPSec), we introduce the IPSec architecture and we look at how it works.

IPSec security services

3.2.1 IPSec Overview

IPSec is the only technique for Internet security that works with all forms of Internet traffic. IPSec runs over the current version of IP, IPv4, and also the next generation of IP, IPv6. In addition, IP can protect any protocol that runs on the top of IP such as TCP, UDP, and ICMP. IPSec is truly the most extensible and complete network security solution. It enables end-to-end security so that every single piece of information sent to or from a computer can be secured. It can also be deployed inside the network to form Virtual Private Networks (VPN) where two distinct and disparate networks become one by connecting them with a tunnel secured by IPSec.

Features of IPSec

The principal feature of IPSec that enables it to support these varied applications is that it can encrypt and/or authenticate all traffic at IP level. Thus, all distributed applications, including remote login, client/server communication, e-mail, file transfer, Web access, and so on, can be secured. IPSec protections are aplied ”above” the IP routing information and ”below” the application data (Fig. 3.2-1). The router in this case treats encrypted packets just as plaintext packets and ignores the extra IPSec headers. The applications never see the IPSec crypto services, i.e. the used IPSec protocols are transparent for them.

Scenario of IPSec deployment

Fig. 3.2-2 is a typical scenario of IPSec usage. A company maintains LANs at dispersed locations. Nonsecure IP traffic is conducted on each LAN. For traffic off site, through some sort of private or public WAN, IPSec protocols are used. These protocols operate in networking devices, such as routers or firewalls, that connect each LAN to the outside world. The IPSec networking device will encrypt and authenticate all traffic going into the WAN, and decrypt and verify the authenticity of the traffic comming from the WAN. These operations are transparent to workstations and servers on the LAN. Secure transmission is also possible with individual users who dial into the WAN. Such user workstations must implement the IPSec protocolls to provide security.

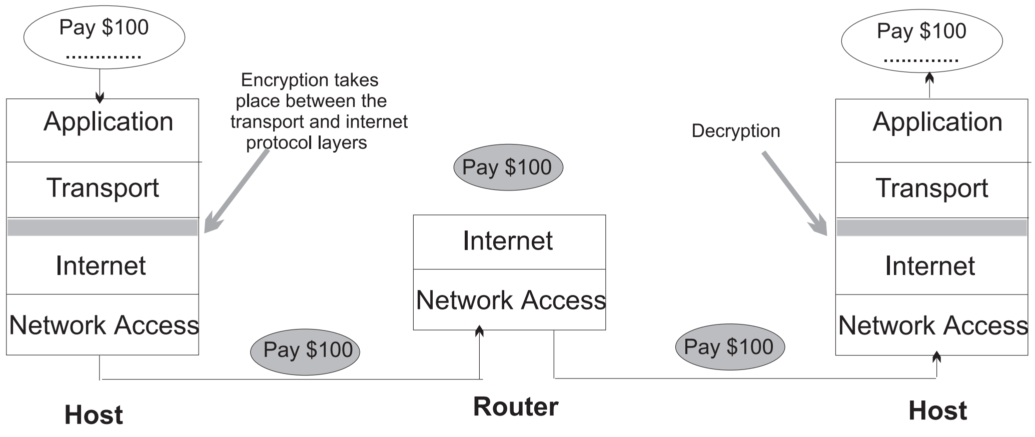


Fig. 3.2-1: Transparency of IPSec in the case of end-to-end encryption

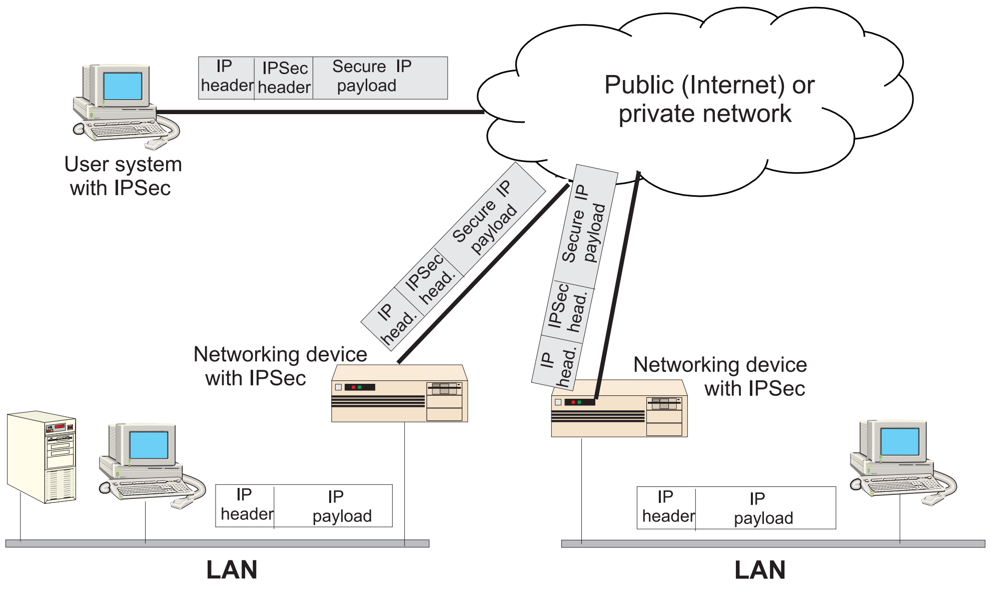


Fig. 3.2-2: An IP security scenario

3.2.2 IPSec Architecture

The IPSec architecture includes various components of IPSec, how they interact with each other, the protocols in the IPSec family, and the modes in which they operate.

RFCs for IPSec

The architecture document for IPSec, RFC 2401, defines the base architecture upon which all implementations are built. It defines the security services provided by IPSec, how and where they can be used, how packets are constructed and processed, and the interaction of IPSec processing with the policy. The IPSec working group at the IETF has published 12 RFCs which define various aspects of IPSec. For example, RFC 1825 gives an overview of a security architecture, RFC 1826 gives a description of a packet authentication extension to IP, RFC 1828 describes the authentication mechanism, RFC 2406 defines ESP (older version is RFC 1827), and RFC 1829 describes the specific encryption mechanism. Support for these features is mandatory for IPv6 and optional for IPv4. In both cases, the **security features are implemented as extension headers that follow the main IP header**. The extension header for authentication is known as authentication header (AH) and the header for encryption is known as Encapsulated Security Payload (ESP) header.

IPSec roadmap

In order to understand, implement and use IPSec, it is necessary to understand the relationship between the building blocks (protocols) in its architecture. The **IPSec roadmap** shown in Fig. 3.2-3 defines how various components of IPSec interact with each other.

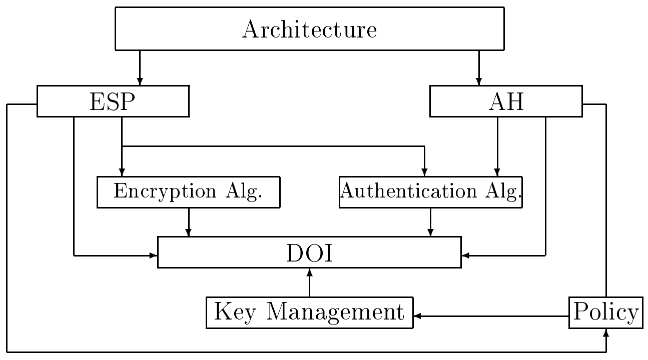


Fig. 3.2-3: IPSec Architecture

Building blocks of IPSec

The IPSec protocol that provides confidentiality is the **Encapsulating Security Payload (ESP)**. Moreover, ESP also provides data integrity, data source authentication of IP Packets, and protection against replay attacks. The **Authentication Header (AH)** is an IPSec protocol that provides everything ESP provides except confidentiality. In order to communicate, each pair of hosts using IPSec must establish **security association (SA)** with one another, i.e. SAs are used to define the processing done on a specific IP packet: what types of protection to apply, how to do encryption and authentication, and which keys need to be used. The SA that applies to a given IPSec header is determined by the packet destination IP address and the **SPI (security parameter index)** field in the packet header. The packets are identified by a SPI, the IPSec protocol value, and the destination adress to which SA applies. SA may be created manually or dynamically and resides in the **Security Association Database (SADB)**.

The IPSec architecture defines the granularity by which a user may specify his policy. This allows for diffeent traffic flows to have different protection. For example, one may specify IPSec policy on a network security gateway that requires all traffic between its local protected subnet and the subnet of remote peer be encrypted with DES and authenticated with HMAC-MD5, while all telnet traffic from the remote subnet to a mail server be encrypted with 3DES and authenticated with HMAC-SHA, and all Web traffic to another server be encrypted with IDEA and authenticated with HMAC-RIPEMD. The IPSec policy is maintained by the **Security Policy Database (SPD)**. Each entry of the SPD defines the traffic to be protected and how to protect it. For each packet entering or leaving the IP stack, the IPSec kernel consults the SPD for the possible application of security.

Prior to an IP packet being secured by IPSec, a SA must exist, which can be, as we mentioned above, manually or dynamically created. The **Internet Key Exchange (IKE)** is used to create them dynamically. IKE negotiates SAs on behalf of IPSec and populates the SADB. IKE is envoked by the kernel or by the policy management application or by another IKE peer to establish keys.

The **Domain of Interpretation (DOI)** block containes values needed for the other documents to relate to each other. These include identifiers for approved encryption and authentication algorithms, as well as operational parameters such as key lifetime.

The IPSec protocols can be used to protect either an entire IP payload or the upper-layer protocols of an IP payload. This distinction is handled by considering two different modes of IPSec (Fig. 3.2-4):

IPSec modes

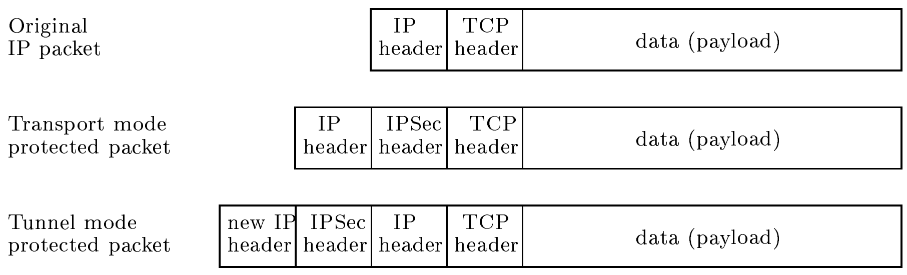


Fig. 3.2-4: IPSec modes

- **Transport mode**. It is used to protect upper-layer protocols (e.g. TCP). An IPSec header is inserted between the IP header and the upperlayer protocol header. This mode can only be used to protect packets where the communications endpoint is also the cryptographic endpoint (end-to-end communication).

Transport mode of IPSec

- **Tunnel mode**. It is used to protect the entire IP packet. To achieve this, after the IPSec fields (AH or ESP or both) are added to the packet, the entire packet plus the security fields is treated as a payload of new ”outer” IP packet with a new, outer IP header. The entire original (inner) packet travels through a ”tunnel” from one point of an IP network to another. No routers along the way are able to examine the inner IP header. Because the original packet is encapsulated, the new, larger packet may have totally different source and destination addresses. This mode can be used in place of transport mode, and in addition may be used by security gateways to provide security services on behalf of other network entities (e.g. VPNs). In this case the communicatons endpoints are those specified in the inner header (which are protected) and the cryptographic endpoints (points where the encryption, decryption or authentication take place) are those of the outer IP header. A security gateway decapsulates the inner IP packet upon the conclusion of IPSec processing and forwards the packet to its ultimate destination.

Tunnel mode of IPSec

When describing the ESP and AH, we will discuss more about both modes.

3.2.3 Encapsulating Security Payload (ESP)

The encapsulating Security payload (ESP) is an IPSec protocol which provides confidentiality, data origin authentication, anti-replay, and data integrity services to IP. It does so by inserting a new header - an ESP header - after the IP header and before the data to be protected, and appending an ESP trailer (Fig. 3.2-5). So, it encapsulates the data it protects. It encrypts the data contents of the reminder of the packet. The format of the ESP varies according to the type and mode of encryption being used. An ESP packet is identified by the protocol field of an IP header. If its value is 50, it is an ESP packet and immediately after the IP header is an ESP header.

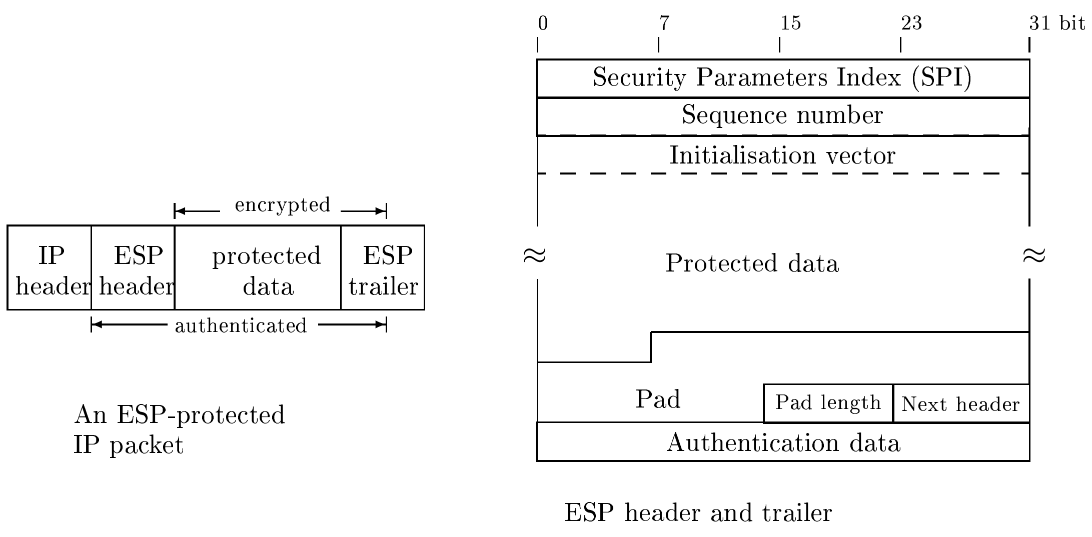


Fig. 3.2-5: ESP is both a header and a trailer - it encapsulates the data it protects

ESP as a generic mechanism

Since ESP provides both confidentiality and authentication, the specific algorithm used by both encryptor and authenticator is determined by the corressponding components in its Security Association (SA). By separating the base ESP definition and format from the actual algorithms which can be used, ESP is generic and extensible security mechanism, i.e many algorithms can be deployed to encrypt or authenticate. The ESP specification imposes requirements on transforms (encryption and authentication algorithms) used with ESP but does not specify what those transforms are. This depends on the specification for implementation.

Encryption algorithms for ESP

The current specification dictates that a compliant implementation must support the DES encryption algorithm in cipher block chaining mode (CBC). A number of other algorithms have been assigned identifiers in the DOI document and can be therefore used for encryption. These include triple DES in CBC mode (3DES-CBC), Blowfish-CBC, RC5-CBC, IDEA-CBC, triple IDEA-CBC or CAST-CBC. All encryption algorithms used in ESP must operate in CBC mode. CBC requires that the amount of data to encrypt be a multiple of the block size of the cipher, and this is here met by padding the data block. When using CBC mode also an initialization vector (IV) is required, which is here contained in the payload field, generally in the first bytes.

Authentication algorithms for ESP

ESP supports the use of a message authentication codes (MAC) with a default length of 96 bits. The current specification dictates that the compilant implementation of the authentication function of ESP, the truncated output of HMAC-MD5 and HMAC-SHA[6] can be used.

ESP Format

Format of the ESP header

The ESP specification defines the format of the ESP header. It immediately follows the IP header, regardless of the mode that ESP is in. After the ESP header follows an upper layer protocol header (e.g. TCP header). The ESP header contains the following fields:

- **Security Parameter Index (SPI)**, 32 bits: This value combined with the destination adress and protocol in the preceding IP header identifies the appropriate Security Association to use in the packet processing. It is an arbitrary number and is selected by the destination during the key exchange. It is authenticated but not encrypted.
- **Sequence number**, 32 bits: This is a unique and monotonically increasing number inserted into the header by the sender. It provides an anty-replay function and is authenticated but not encrypted.

The actual data being protected (encrypted) by ESP is contained in the **playload data field**. Therefore, the length of this field depends on the length of the data. It is a transport layer segment (transport mode) or IP packet (tunnel mode). This data is also used to contain any initialisation vector that the encryption algorithm may require. The**ESP trailer** contains the following fields:

Format of the ESP trailer

- **Padding** (if any, 0-255 bits): Certain modes of encryption algorithms require that the input to the cipher be a multiple of its block size. Padding accomplishes this.
- **Pad length** (8 bits): Defines how much pad has been added, so that the recepient can restore the actual length of the payload data.
- **Next header** (8 bits): Indicates the type of data that is contained in the payload data field. If ESP is applied in tunnel mode, this will be value four, indicating IP-in-IP. If ESP is applied in thansport mode, this value will indicate the type of upper-level protocol that follows (e.g. TCP would be six).

6 HMAC is a special kind of keyed hash, with benifit that its generation is much faster than a digital signature. HMAC can be utilized with any existing hash function, so SHA becomes HMAC-SHA and MD5 becomes HMAC-MD5. The HMAC construction is cryptographically stronger that the underlying hash function. HMAC uses two pad values (an inner pad ipad and an outher pad opad) to maintain state of the HMAC. The HMAC based on hash algorithm H of message M using key K is defined as

$$HMAC(K, M) = H(KXORopad, H(KXORipad, M)).$$

All message authentication done in IPSec uses HMACs.

- **Authentication data** (variable length): Holds the result of the data integrity check done on the ESP packet (keyed hash function computed over the ESP packet minus the authentication data field). The size of this field depends on authentication algorithm employed by the SA to process this packet.

Protection of different packet fields

The ESP header is not encrypted, but a portion of the ESP trailer is. The data that the recepient needs to process the packet must be in clear text. Since the SPI is used, along with the destination IP adress of the IP header of this packet, to identify an SA it must be in clear text. In addition, the sequence number and the authentication data are also in clear text. It is due to the specified **order of processing** of IP packets: First verify the sequence number, then verify the integrity of the data, then decrypt the data. Since decryption is last, the sequence number and authentication data must be known.

ESP in transport mode

Depending on what actually ESP is protecting, the ESP header is applied to an IP packet in transport mode or in tunnel mode. Transport mode is used to encrypt and optionally authenticate the data carried by IP (e.g. a TCP segment). This mode provides confidentiality for any application that uses it, thus avoiding the need to implement confidentiality in every individual application. It adds little overhead to the total length of the IP packet (advantage), but in this mode it is possible to do traffic analysis on the transmitted packets. In transport mode an ESP header is inserted between the IP header and the upper-layer protocol header of an IP packet, and an ESP trailer is placed after the IP packet. If authentication is selected, the ESP Authentication data field is added after the ESP trailer. The entire transport level segment plus the ESP trailer are encrypted (see Fig. 3.2-6 b).

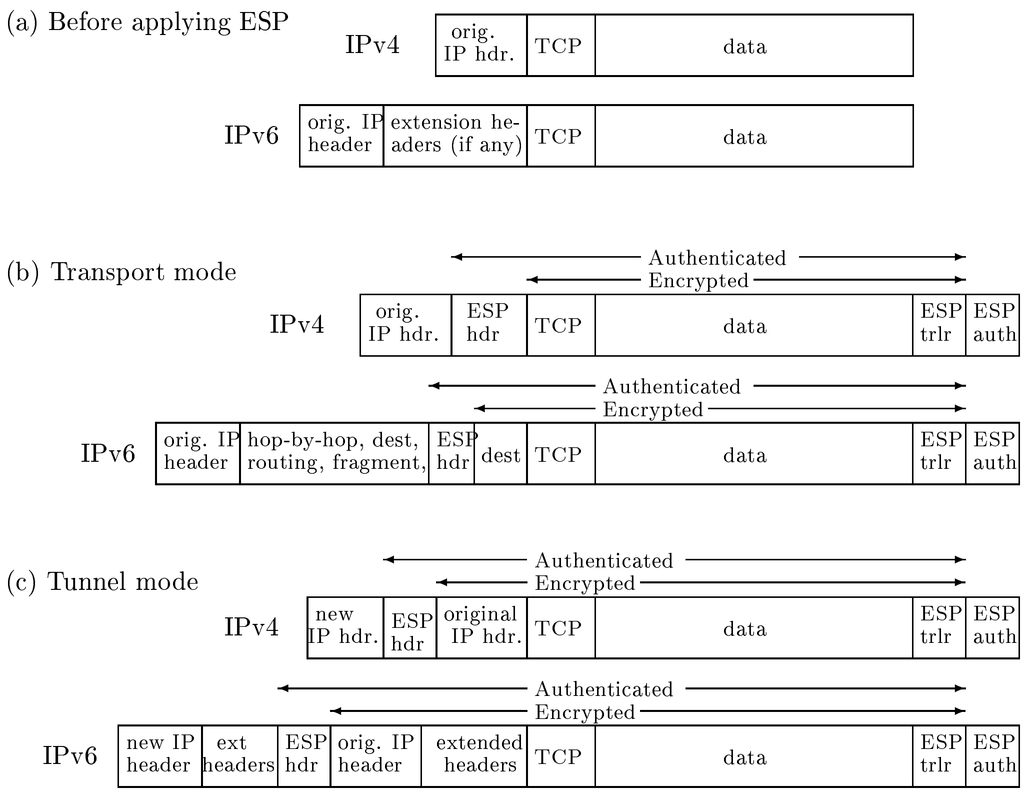


Fig. 3.2-6: Scope of ESP encryption and authentication

In tunnel mode the entire protected IP packet is encapsulated in the ESP header and a new IP header is added to that. For this mode, the ESP header is prefixed to the packet and then the packet plus the ESP trailer are encrypted. Because the IP header contains the destination address and possibly source routing directives, it is not possible simply to transmit the encrypted IP packet prefixed by the ESP header. Intermediate routers would be unable to process such a packet. Therefore, it is necessary to encapsulate the entire block (ESP header plus ciphertext plus authentication data, if present) with a new IP header that will contain sufficient information for routing, but not for traffic analysis (see Fig. 3.2-6 c.). So, this method can be used to counter traffic analysis.

3.2.4 Authentication Header (AH)

The AH protocol, which provides support for data integrity and authentication of IP packets, is defined in RFC 2402. It defines the method of protection, the placement of the header, the authentication coverage, the input and output processing rules, but it does not define the authentication algorithm to use (i.e. it is generic and extensible).

Features of AH

Just like ESP, AH can be used to protect the upper-layer protocol (transport mode) or an entire IP datagram (tunnel mode). In either case the AH header immediately follows an IP header. AH is an IP protocol and an AH-protected IP packet is just another IP packet. Therefore, AH can be used alone or in conjunction with ESP. The data integrity that AH provides is subtly different from that provided by ESP; **AH authenticates portions of the outer IP header.**

AH header format

When AH and ESP are protecting the same data, the AH header is always inserted after the ESP header. The AH header is much simpler than the ESP header, because it does not provide confidentiality (there is no initialisation vector and no trailer as there is no need for padding and pad length indicator). As shown in Fig. 3.2-7, the AH header has the following fields:

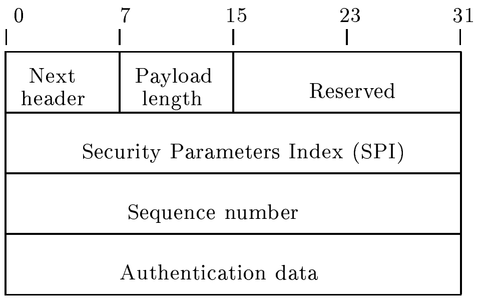


Fig. 3.2-7: The AH header

- **Next header** (8 bits): It indicates what follows the AH header. In transport mode it will be the value of the upper-layer protocol being protected (TCP or UDP) and in tunnel mode it will be the value 4 (for IPv4 IP-in-IP encapsulation) or the value 41 (for IPv6 encapsulation).

- **Payload length** (8 bits): This field indicates the length of the header itself in 32-bit words minus 2.
- **Reserved field**: It is not used and must be set to zero.
- **Security Parameters Index (SPI)**, 32 bits: Identifies a security association used to authenticate the packet (together with the destination adress of the outer IP header).
- **Sequence number** (32 bits): It is used to provide the anti-replay function and is monotonically increasing counter identical to that used in ESP.
- **Authentication data** (variable): This field contains the result of the integrity checking function. AH does not define authenticator, but HMACSHA- 96 and HMAC-MD5-96 are mandatory to implement.

Since authentication is based on the use of HMACs, the two parties must share a key, which is established with the Internet Key Exchange Protocol (IKE). The HMAC is calculated over the following:

Calculating the authentication data (HMAC)

- IP header fields that either do not change in transit (immutable) or that are predictable. Fields that may change in transit are set to zero for purposes of HMAC calculation at both source and destination.
- The AH header except the authentication data field (this is set to zero for purposes of calculation at both source and destination).
- The entire upper-level protocol data, which is assumed to be immutable in transit (e.g a TCP segment or an inner IP packet in tunnel mode).

For IPv4, examples of immutable fields are Internet Header Length and Source Adress. An example of an muttable but predictable field is the Destination Address. Examples of mutable fields that are zeroed prior to the HMAC calculation are the Time to Live and Header Checksum Fields. Note that both source and destination adress fields are protected, so that adress spoofing[7] is protected. For IPv6, examples in the base header are Version (immutable), Destination Adress (mutable but predictable), and Flow Label (mutable and zeroed for the HMAC calculation).

7 In IP Spoofing an intruder creates packets with false IP adresses

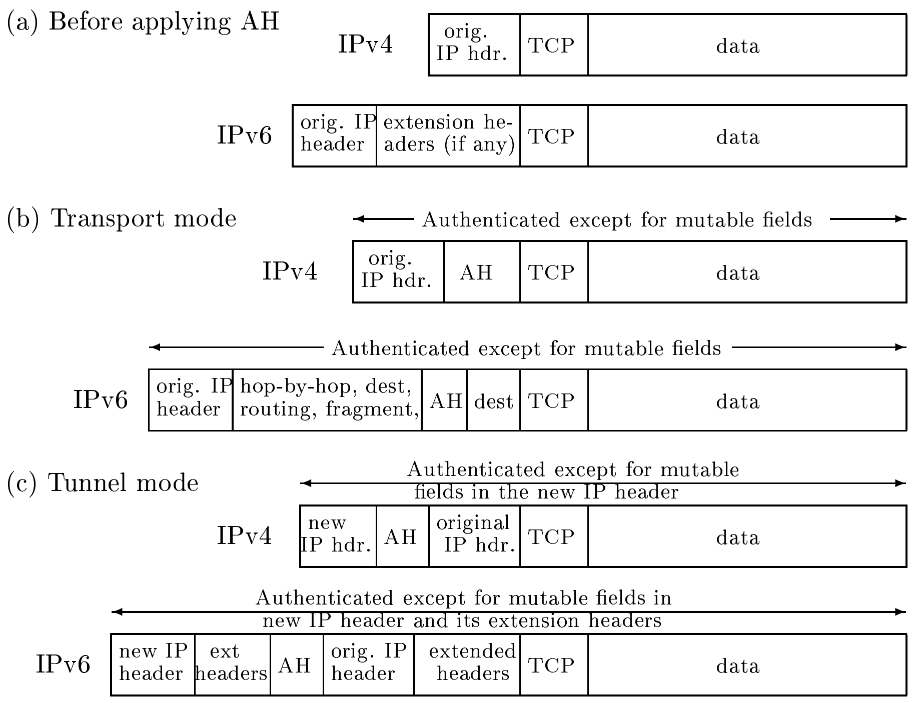

(a) Before applying AH
IPv4
orig. IP hdr.
TCP
data
IPv6
orig. IP header
extension headers (if any)
TCP
data
(b) Transport mode
Authenticated except for mutable fields
IPv4
orig. IP hdr.
AH
TCP
data
Authenticated except for mutable fields
IPv6
orig. IP header
hop-by-hop, dest, routing, fragment,
AH
dest
TCP
data
(c) Tunnel mode
Authenticated except for mutable fields in the new IP header
IPv4
new IP hdr.
AH
original IP hdr.
TCP
data
Authenticated except for mutable fields in new IP header and its extension headers
IPv6
new IP header
ext headers
AH
orig. IP header
extended headers
TCP
data

Fig. 3.2-8: Scope of IPSec authentication

Ways of using IPSec authentication

The IPSec authenticatein can be used in two ways. **End-to-end authentication** is when authentication is provided directly between a server and a client workstation (which can be either on the same network as the server or on an external network), and this case uses transport mode SA. **End-to-intermediate authentication** is the case when a remote workstation authentificates itself to the corporate firewall, either for the access to the entire internal network or because the requested server does not support the authentication feature. This case uses tunnel mode SA.

Scope of authentication provided by AH

We now look at the scope of authentication provided by AH and the AH location for the two modes. For **transport mode AH** using IPv4, the AH is inserted after the original IP header and before the IP payload, e.g. a TCP segment (see Fig. 3.2-8 b). In the context of IPv6, AH is viewed as an end-toend payload; that is, it is not examined or processed by intermediate routers. Therefore, the AH appears after the IPv6 base header and the hop-by-hop, routing, and fragment extension headers.

For **tunnel mode AH**, the entire original IP packet is authenticated, and the AH is inserted between the original IP header and a new outer IP header (see Fig. 3.2-8 c). The inner IP header carries the ultimate source and destination adresses, while an outer IP header may contain different IP adresses (e.g. adresses of firewalls or other security gateways). With tunnel mode, the entire IP packet, including the entire inner IP header (and, in the case of IPv6, the outer IP extension headers) is protected except for mutable and unpredictable fields.

3.2.5 Security Association (SA)

Definition of SA

A key concept that appears in both the authentication and confidentiality mechanisms of IP is the Security Association (SA). An association is a one-way relationship between a sender and a receiver that offers security services to the traffic carried on it. If a peer relation for two-way secure exchange is needed, then two SAs are requird. A SA is uniquelly defined with three parameters: **Security Parameters Index** (SPI, a bit string which is carried in AH and ESP headers to enable the receiving system to select the SA under which a received packet will be processed), **IP destination address** (address of the destination endpoint od the SA, which may be an end user system or a network system such as firewall or router), and **Security Protocol Identifyer** (which indicates whether the association is an AH or ESP security association. Hence, in any IP packet the SA is uniquely identified by the destination adress in the IPv4 or IPv6 header and the SPI in the enclosed extension header (AH or ESP). Actually, SA is an abstraction of a security policy and a key.

SADB

In each IPSec implementation, there is a nominal (depends on the implementor) **Security Association Database (SADB)** that defines the parameters associated with each SA. An entry in this database (i.e. each SA) is defined with several parameters and here we will only breafly mention some of them: sequence number counter, sequence counter overflow, anti-replay window, AH information (authentication algorithm, keys, key lifetimes), ESP information (encryption and authentication algorithm, keys, initialisation values, key lifetimes), lifetime of this SA (interval after which an SA must be replacer with a new SA or terminated), IPSec protocol mode (tunnel or transport), and others.

SA bundle

An individual SA can implement either the AH or ESP protocol, but not both. Sometimes, the particular traffic flow will call for the services provided by both ESP and AH. Further, a particular traffic flow may require IPSec services between hosts and, for the same flow, separate services between security gateways, such as firewalls. In all of these cases, multiple SAs must be employed for the same traffic flow to achieve the desired IPSec services. The term **security association bundle** refers to a sequence of SAs through which traffic must be processed to provide a desired set of IP services. The SAs in the bundle may terminate at different endpoints or at the same endpoints. SAs may be combined into bundles in two ways:

Combining of SAs

- Transport adjacency: Reffers to applying more than one security protocol to the same IP packet, without invoking tunelling.
- Iterated tunelling: Reffers to the application of multiple layers of security protocols affected through IP tunelling. This approach allows for multiple levels of nesting, since each tunnel can originate or terminate at a different IPSec site along the path.

The two approaches can be combined in a number of ways to yield the desired user configuration. The basic combinations of SAs are given in Fig. 3.2-9.

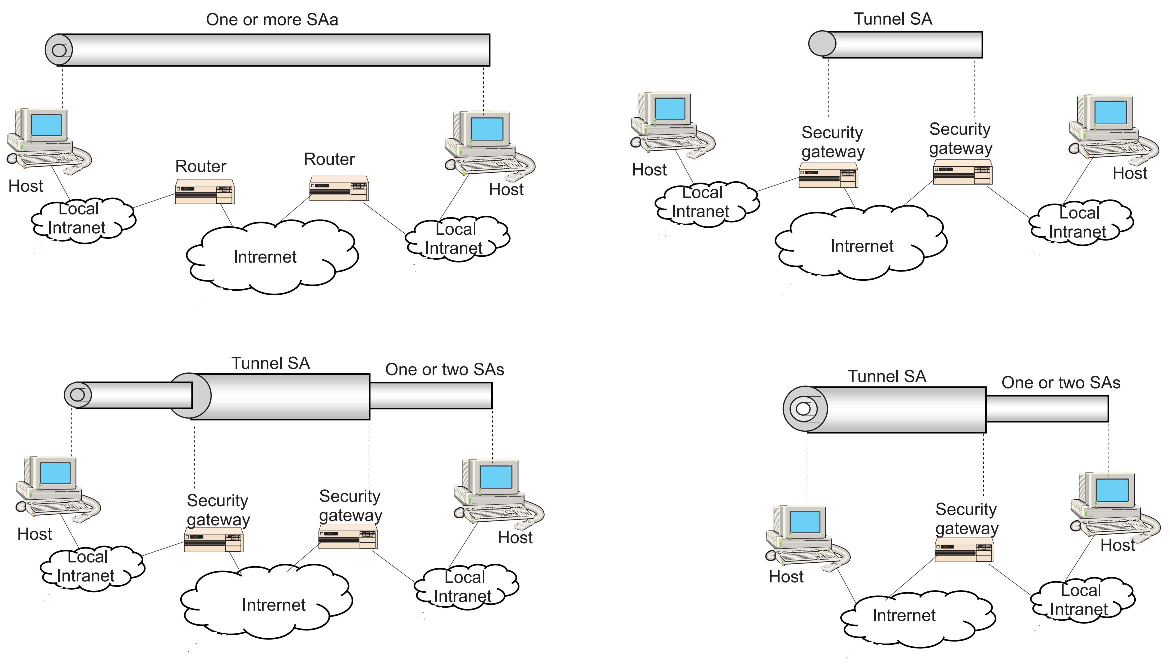


Fig. 3.2-9: Basic combinations of security associations

3.2.6 IPSec Key Management

As mentioned above, Security Associations are used with IPSec to define the processing done on a specific IP packet. An outbound packet produces a hit in the Security Policy Database (SPD) and the SPD entry points to one or more SAs (SA bundle). If there is no SA that instantiates the policy from the SPD, it is necessary to create one. This is where the **Internet Key Exchange (IKE)** is used to create one. Actually, the IKE is the key management system of IPSec - Its purpose is to establish shared security parameters and authenticated keys (in other words, Security Associations) between IPSec peers, i.e. it defines the steps two peers must take to establish a shared, authenticated key. Actually, IKE is general purpose security exchange protocol and may be used for policy negotiation and establishment of authenticated keying material for a variety of needs.

Definition of IKE

The IPSec architecture document mandates support of two types of key management:

Types of key management in IPSec

- Manual: A system administrator manually configures each system with its own keys and with the keys of other communicating systems. Manual key management is available in any compilant IPSec implementation, but is only practical for small, relatively static environments.

- Automated: An automated system enables on-demand creation of keys for SAs and facilities the use of keys in a large distributed system.

The default automated key management protocol for IPSec is referred to as **ISAKMP/Oakley** and consist of Oakley key determination protocol and the Internet Security Association and Key Management Protocol (ISAKMP).

Features of Oakley

Oakley is a key exchange protocol based on the Diffie-Hellman algorithm which provides added security on it. It is a generic protocol in that it does not dictate specific formats. Oakley is designed to retain the advantages of Diffie- Hellman (secret keys are designed only when needed and there is no need to store secret keys for a long period of time, exposing them to increased vulnerability and the key exchange does not require any preexisting infrastructure) while countering its weaknesses (does not provide any information about the identities of the parties, is subject to man-in-the-middle attack, and is computationally intensive). Oakley supports the use of different groups for the Diffie-Hellman key exchange. The current specification includes the following groups: Modular exponentiation with 768-bit modulus and fixed parameters, modular exponentiation with 1024-bit modulus and fixed parameters, modular exponentiation with 1024-bit modulus and variable parameters (to be determined), elliptic curve group over 2^{155} with fixed parameters, and elliptic curve group over 2^{185} with fixed parameters[8]. Oakly also employs mechanism which protects against replay-attacks and enables use of three different types of authentication methods: digital signatures (the key exchange is authenticated with signing a mutually obtainable hash), public-key encryption (the key exchange is authenticated by encrypting parameters), and symmetric-key encryption. The Oakley specification includes a number of key exchanges that are allowable under the protocol.

Features of ISAKMP

The **Internet Security Association and Key Management Protocol (ISAKMP)**defines procedures and packet formats to establish, negotiate, modify, and delete Security Associations. As a part of SA establishment, ISAKMP defines payload formats for exchanging key generation and authentication data. These payload formats provide a consistent framework independent of the specific key exchange protocol, encryption algorithm and authentication mechanism.

Payload types in an ISAKMP message

An ISAKMP message consists of an ISAKMP **header** followed by one or more ISAKMP **payloads**. The ISAKMP header contains information about the type and the length of the payload that follows. For ISAKMP, the following payload types are defined:

- SA payload: Used to negotiate security attributes,
- Proposal payload: Indicates protocol to be used (ESP or AH) in this SA and number of transforms,
- Transform payload: Defines a security transform to be used to secure the communication channel for the designed protocol (e.g 3DES, HMACSHA),
- Key exchange payload: Supports a variety of key exchange techniques,
- Identification payload: Used to exchange identification information,
- Certificate payload: Used to transport public-key certificates,

8 For more information see [KCL+00], chapter 4.

- Certificate request payload: Used from sender to request certificate of the other communicating entity,
- Hash payload: Contains data generated by a hash function over some part of the message,
- Signature payload: Contains data generated by a digital signature function over some part of the message,
- Nonce payload: Contains random data used to protect against replayattack,
- Notification payload: Used to transmit notification data, such as an error or status information, and
- Delete payload: Indicates SAs that the sender has deleted fom its database and which are therefore no longer valid.

So, ISAKMP provides a framework for message exchange, with the payload types mentioned aboved serving as building blocks. To resume, ISAKMP defines how two peers communicate, how the messages they use to communicate are constructed, and also defines the state transitions they go through to secure their communication.

The IKE at work

The IKE (which is actually a hybrid protocol of Oakley and ISAKMP) is envoked by the kernel or by the policy management application or by another IKE peer to establish keys. The IKE protocols are performed by each party that will be performing IPSec; the IKE peer is also IPSec peer. The protocol is of **request-response type** with an initiator and a responder[9]. The initiator is the party that is instructed by IPSec to establish some SAs as a result of an outbound packet matching an SPD entry. It initiates the protocol to the responder.

Implementation of compliant IKE requires adherence to three documents: the base ISAKMP (specification (RFC 2408), the Domain of Interpretation for IPSec (RFC 2407), and the IKE specification itself (RFC 2409).

Note that the key management mechanism is coupled to the authentication and privacy mechanisms only by the way of Security Parameter Index. Hence, authentication and privacy have been specified independent of any specific key management system.

3.2.7 IPSec Policy

Definition of policy

The IPSec policy determines the security services offered to a packet and the treatment of the packet in the network. It is the security interface between the human and the computer and therefore extremely important. The transition from human to computer involves policy definition, policy representation, policy management and the interactions between the policy and the various components of an IPSec

9 Since the protocol is pretty complicated, we are not going to discuss it here in detail. For more information see [DH99].

system (IKE, SADB, SPD and IPSec process itself). Policy is not a standard. IETF does not mandate any particular representation of policy nor does it enforce any particular implementation - the policy issues are left to implementations.

IPSec allows very complex policy definitions. Because of its notions of selectors, the policy rules applied to inbound and outbound packets is very rich. Selectors can be gross - an entire subnet - or fine - a specific port and protocol to a specific host.

Requirements on a policy system

There are several requirements imposed on any policy system. It must be possible to specify security policy for each individual host. The policy may reside on the host or it may reside on the router or firewall in the network. The policy may specify end-to-end security (policy between two hosts) or it may specify policy for this host to another network. It also must be possible to specify security policy for an entire network. The network policy may take the form of a network and subnet, or the form of a range of adresses. Important is that a uniform security policy can be assigned to all entities of the network. It must also be possible to specify security policy to the granularity of port, protocol and application. For example, it must be possible to protect all telnet (tcp port 23) trafic destined for a particular subnet. Moreover, there must be an ability to specify multiple protocols per flow. A flow is always identified by the tuple <src, dst, src port, dst port, protocol>. For each flow it must be possible to specify the desired action: **drop** (reject the packets), **protect** (encrypt and/or authenticate the packets), or **pass** (transmit the packets without any security action). It must be also possible to configure tunnels of more than one level deep (nested tunnels). This is usually a result of combining various distinct policy requirements into a single unified network policy. Different IKE identities (IP adresses or domain names) for access and authentication must also be supported. For example, a node can be configured to accept/reject connections by a particular user or by a set of users belonging to a particular domain. It must also be possible to define a policy such that manually-keyed SAs are used instead of IKE in order to protect certain traffic (i.e. to disable the automated key management). Multiple security options must be supported too. IKE provides the ability to offer multiple choices during negotiation so that the two peers can agree on something that both support. For example, host A may want to use ESP and 3DES with nodes that implement 3DES, but with nodes that do not support 3DES it may choose to communicate using DES instead of dropping the connection.

Policy representation

The **policy representation** is a database problem and involves two aspects:

- The physical representation: It defines the format in which the policy is represented, where it is stored, and how it is updated and modified.
- The interface: It defines how the various IPSec components acquire and manage a policy.

The physical representation of a policy depends of the distribution mechanism. Because of its flexibility and generic design LDAP[10] (Lightweight Directory Access Protocol) has received the most attention as a mechanism to distribute policy. The policy is stored centrally in a policy server, which is responsible for maintaining the policy for all nodes in the domain. The policy is either downloaded into the various nodes of the network or is fetched dynamically by the nodes.

3.2.8 IPSec Implementation

Ways of implementation

IPSec may be implemented in end systems or in security gateways such as routers and firewalls. Typically this is done by directly modifying the IP stack. When access to the IP stack of the machine is not possible, IPSec may be implemented as a shim that extracts and inserts packets from the IP stack, called **”Bump in the Stack” (BITS)**. IPSec can also be implemented as external, dedicated crypto device, generally called **”Bump in the Wire” (BITW)** that may be independently addressable.

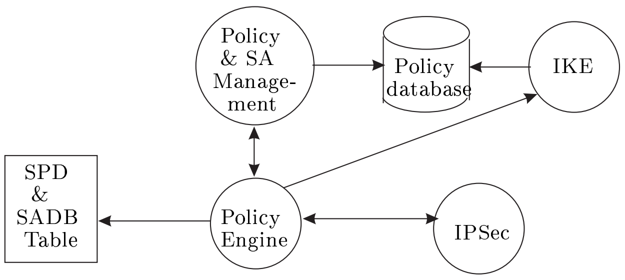


Fig. 3.2-10: IPSec implementation architecture

Interaction among the IPSec components

Fig. 3.2-10 indicates the various components of IPSec and the interaction among them. We now briefly summarize the above discussed IPSec components:

- IPSec base protocols: This componentt implement ESP and AH, which interact closely with the network and the transport layer. In fact, IPSec base protocol is part of the network (IP) layer. ESP and AH process the headers, interact with SPD and SADB to determine the security that is offered to the packet, and handles network layer issues such as fragmentation.
- SPD: It is consulted for both outbound and inbound processing of the packet, because it determines the security afforded to the packet.
- SADB: Maintains the list of active SAs for outbound and inbound processing. The SADB is populated with SA either manually or via an authomatic key management system such as IKE. The choice of the data structure to store SPD and SADB is very crucial to the performance of IPSec processing. This depends on the performance requirements and the number of entries in SADB and SPD.

10 LDAP is an advantageous mechanism which provides a simple method of depositing and retrieving policy from a central repository. Describing LDAP in detail is out of the scope of this course.

- IKE: The Internet Key Exchange is envoked by the policy engine when the policy mandates an SA, or when SA(s) is to be established. IKE is also invoked by its peer when the node needs to communicate securely. It is responsible for dynamically populating the SADB and is in a quiescent state until its services are needed. It also provides an interface to the SPD to inform it about establishing new SAs. It is normally a user-level process in the operating system.
- Policy and SA management: These are applications that manage the policy and SA. This module is implemented at the user level. The user interacts with this module for all policy-related processing. This module is responsible for adding, looking up and deleting policies and for manually adding, creating and deleting SAs.

3.2.9 IPSec Protocol Processing: An Example

Classification of the IPSec processing

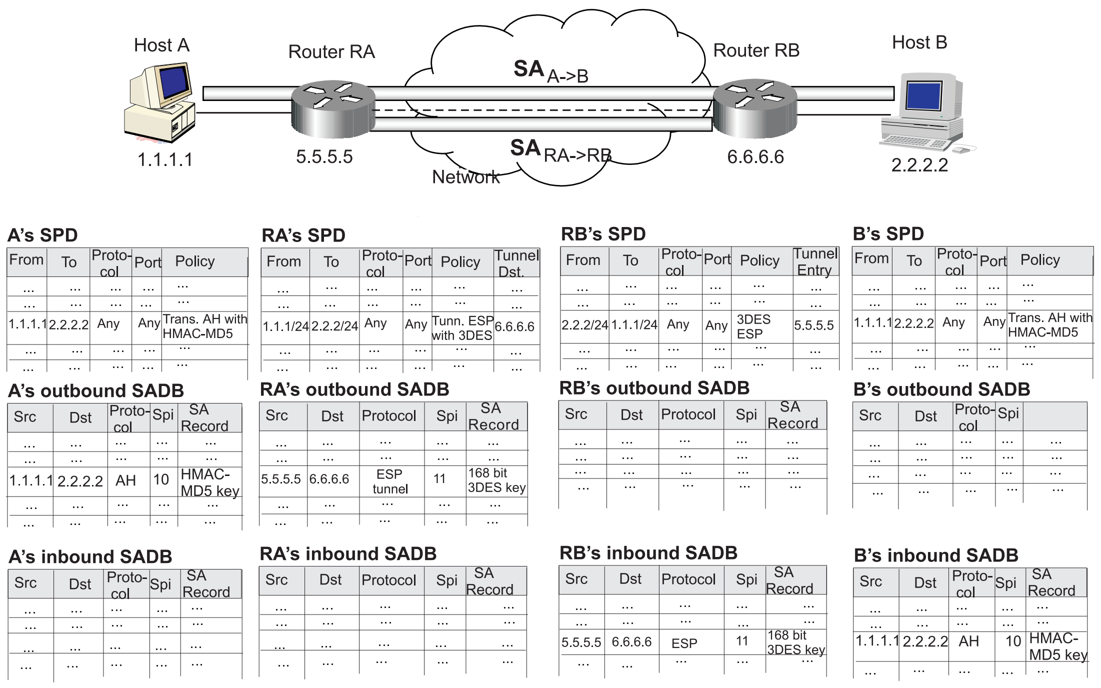
The IPSec processing is broadly classified into outbound versus inbound processing, and AH versus ESP (and their variants) processing. The input (inbound) and outgoing (outbound) packets of a node are differently processed. Moreover, the transform and header processing is different between AH and ESP. In the outbound processing a IPSec header is constructed and some operations (e.g. encryption) are performed. Since in the inbound processing only the existing header is checked, it is simpler than the outbound processing. Before we sketch the inbound and outbound prcessing of packets on an exapmle, we summarize the functionalities of IPSec in transport and tunnel mode in Fig. 3.2-11.

	Transport mode SA	Tunnel mode SA
AH	Authenticates IP payload and selected portions of IP header and IPv6 extension headers.	Authenticates entire inner IP packet (inner header plus IP payload) plus selected portions of outer IP header and outer IPv6 extension headers.
ESP	Encrypts IP payload and any IPv6 extension headers following the ESP header.	Encrypts inner IP packet.
ESP with authentication	Encrypts IP payload and any IPv6 extension headers following the ESP header. Authenticates IP payload but not IP header.	Encrypts inner IP packet. Authenticates inner IP packet.

Fig. 3.2-11: Transport mode and tunnel mode functionality

Outbound Processing

To explain the outbound processing, we now consider the example shown in Fig. 3.2-12. Suppose, we have an HTTP packet (TCP, port 80) generated by host *A* and destined to host *B* (a Web server) traversing routers *RA* and *RB*. The policy on *A* for packets destined to host *B* mandates using AH in transport mode with HMAC-MD5. The policy on router *RA* mandates that all packets destined to the network 2.2.2/24 be encrypted with ESP using 3DES and tunneled to *RB*.



A's SPD

From	To	Proto-col	Port	Policy
...	...	...	...	...
...	...	...	...	...
1.1.1.1	2.2.2.2	Any	Any	Trans. AH with HMAC-MD5
...	...	...	...	...
...	...	...	...	...

RA's SPD

From	To	Proto-col	Port	Policy	Tunnel Dst.
...	...	...	...	...	...
...	...	...	...	...	...
1.1.1/24	2.2.2/24	Any	Any	Tunn. ESP with 3DES	6.6.6.6
...	...	...	...	...	...
...	...	...	...	...	...

RB's SPD

From	To	Proto-col	Port	Policy	Tunnel Entry
...	...	...	...	...	...
...	...	...	...	...	...
2.2.2/24	1.1.1/24	Any	Any	3DES ESP	5.5.5.5
...	...	...	...	...	...
...	...	...	...	...	...

B's SPD

From	To	Proto-col	Port	Policy
...	...	...	...	...
...	...	...	...	...
1.1.1.1	2.2.2.2	Any	Any	Trans. AH with HMAC-MD5
...	...	...	...	...
...	...	...	...	...

A's outbound SADB

Src	Dst	Proto-col	Spi	SA Record
...	...	...	...	...
...	...	...	...	...
1.1.1.1	2.2.2.2	AH	10	HMAC-MD5 key
...	...	...	...	...
...	...	...	...	...

RA's outbound SADB

Src	Dst	Protocol	Spi	SA Record
...	...	...	...	...
...	...	...	...	...
5.5.5.5	6.6.6.6	ESP tunnel	11	168 bit 3DES key
...	...	...	...	...
...	...	...	...	...

RB's outbound SADB

Src	Dst	Protocol	Spi	SA Record
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

B's outbound SADB

Src	Dst	Proto-col	Spi	
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

A's inbound SADB

Src	Dst	Proto-col	Spi	SA Record
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

RA's inbound SADB

Src	Dst	Protocol	Spi	SA Record
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

RB's inbound SADB

Src	Dst	Protocol	Spi	SA Record
...	...	...	...	...
...	...	...	...	...
5.5.5.5	6.6.6.6	ESP	11	168 bit 3DES key
...	...	...	...	...

B's inbound SADB

Src	Dst	Proto-col	Spi	SA Record
...	...	...	...	...
...	...	...	...	...
1.1.1.1	2.2.2.2	AH	10	HMAC-MD5 key
...	...	...	...	...

Fig. 3.2-12: Example of IPSec processing

Steps in the outbound processing

In most TCP/IP implementations, the transport layer invokes the IP layer to send a packet by calling the function ip output. At the entry point to the ip output, the policy engine is consulted to determine the security services offered to a packet, so **SPD processing** is done at first. The input to the policy engine in our example will be the touple <1.1.1.1, 2.2.2.2, TCP, 80>. The policy engine determines the security services and:

- Drops the packet, if the policy indicates that the packet has to be dropped. In this case the ip output function is expected to discard the packet.
- Transmits the packet in clear text, if the policy indicates that this packet can be transmitted without any extra security processing.
- If the policy indicates that the packet needs security, the policy engine checks if the SAs are already established. If SAs are not already established, the security policy notifies IKE to establish the required SA(s). The packet is queued until the SAs are established.

In our example the policy engine determines that the packet needs to be secured using transport mode AH HMAC-MD5. It also determines that the SA is already established, and hence does not invoke IKE to establish a new SA.

If SA is not established, then **IKE processing** must be done to establish SAs. IKE requires special processing: It is a requirement that all implementations recognize IKE packets and process them without securing them. This is possible as IKE always uses a well-known port (500) and a well-known protocol (UDP).

The next step in the outbound packet processing is the **SA processing**. The particular SA is fetched from the SA Database (SADB). In our example, the HMAC-MD5 key for authentication is read from A's outbound SADB. After the SA processing, the protocol specific component is invoked : transport mode (for both AH and ESP) or tunnel mode (for both AH and ESP). So, at this place we have four possibilities: transport mode ESP processing, transport mode AH processing, tunnel mode ESP processing, and tunnel mode AH processing.

Transport mode AH processing in host A

In our example, transport mode AH processing has to be done: The AH is constructed and appended to the packet, the HMAC-MD5 is computed and placed in the AH authentication data field. The **transport mode AH processing** expects part of the IP header to be constructed before it calculates the hash (the AH header has to be inserted in the middle of a packet). The ideal place to perform AH processing is just before the fragmentation check is performed. The rules for calculating the AH header were described in Section 3.2.4.

The transport mode ESP involves encrypting the transport payload and calculating the hash. The transport mode ESP header is constructed before the IP header is added to the payload. After the ESP transform is applied to the transport payload and the ESP header is added, the ESP packet is forwarded to the IP layer for the IP processing.

Tunnel mode ESP processing in the router RA

The **tunnel mode processing** involves construction of an extra IP header. Tunnels can be nested and can be of any depth as long as they are nested properly. If multiple tunnels have to be constructed from the same node, multiple headers have to be processed. In our example, when router RA receives the packet, the output of the policy lookup in its SPD is a pointer to the Security proAssociation between *RA* and *RB* ($SA_{RA->RB}$). The $SA_{RA->RB}$ indicates that the packet should be protected by a tunnel mode ESP with 3DES. The source and destination fields in the SA record are the source and destination values in the outer IP header. The router *RA* encrypts the data payload, constructs a tunnel mode ESP header and forwards the packet to the IP layer for IP processing. Tunnel mode processing on the hosts is different from that of the router because the host needs to add two IP headers and not one.

After IPSec processing, the packet is passed back to the IP layer. After the IP layer adds the IP header, the packet is forwarded to the data link layer (i.e. network access layer).

Inbound Processing

Because header checking is less complicated than header construction, the inbound processing is simpler than the outbound processing. During inbound processing there is also no interaction with the key management system. The IP layer is invoked by layer 2 (see Fig. 3.2-1) to process the packet that was received over an interface. Most implementations have a function for each protocol that the IP layer can call to process the packet. The IP Layer strips the IP header and invokes the IPSec layer with either an AH or ESP header at the beginning of the packet.

Let us consider the network from Fig. 3.2-12 again, this time from the perspective of inbound processing. In this figure, the SPD and SADB for the receivers (router *RB* and host *B*) are also shown (note that the SPD entries are symmetric).

Tunnel mode ESP processing in the router RB

In our example, *RB* receives a tunneled packet from *RA*. In the case of tunneling, the IPSec layer has to perform an extra check to determine if the SA that was used to process the packet was in fact established to process the packet from the actual source. This is achieved by using the inner-header destination address for SPD lookup. Then IPSec invokes the upper layer to process the packet. In the case of tunneled packets, the upper layer is the IP layer itself. *RB* receives a packet from source 5.5.5.5 with tunneled ESP using 3DES using an SPI of value 11. The lookup in the SADB yields an SA pointer. However, when the policy engine is envoked, the source and the destination address will be that of the inner IP header. The values in this case are 1.1.1.1 and 2.2.2.2. The lookup in the SPD matches the entry whose ”from” and ”to” fields are network prefixes 2.2.2/24 and 1.1.1/24. They also indicate that the packet was tunneled by 5.5.5.5. As the security services afforded to the packet match to that in the SPD, the tunnel is removed (the tunnel header is removed the data is decrypted) and the packet is forwarded to the actual destination (host *B*).

Transport mode AH processing in host B

As the packet comes to host *B*, the IPSec layer extracts the SPI from the AH or ESP header (here 10), and the source and destination address (1.1.1.1 and 2.2.2.2) and protocol from the IP header. The IPSec component then fetches the SA from the SADB using the destination (2.2.2.2), protocol (AH), and SPI (10), and the packet is processed according to the rules defined in ESP and AH sections (if the SADB does not find the SA, an error is logged and the packet is dropped). The policy corresponding to the packet is checked to determine if the IPSec processing is applied appropriately. In the example, the SPD has an entry that specifies that any packets from 1.1.1.1 should have AH in transport mode using HMAC-MD5. The policy engine checks if this is true. In this case, this is true, and hence the packet is accepted. Then, the packet is authenticated and/or encrypted (in our example, it is only authenticated, i.e. *B* computes HMAC-MD5 and compares it with the value in the AH of the packet).

For non IPSec packets, the processing is limited to looking up into the SPD and confirming that the packet can be admitted without any security.

Note that the throughputs of IPSec implementations usually decrease with the number of SAs that are established and maintained simultaneously.

3.2.10 Deployment Examples of Securing Network Traffic with IPSec

IPSec is a robust and extensible mechanism for securing IP datagrams and is ideal for protecting any type of traffic that can travel on the top of IP. By providing security at the IP layer, IPSec allows any application to take full advantage of its functionality. Security is done in one place, in the stack, instead of in each application that requires security (in socket-based security, such as SSL, every application that desires security must be modified). By IPSec all applications can be secured just by modifying the protocol stack. By placing IPSec-enabled hardware at different points in the network (routers, firewalls, hosts) different security deployments can be realized. We now briefly consider some of them.

End-to-End Security

End-to-end security with IPSec

When IPSec exist on a host or endsystem, IP packets can be secured from the data source to the data sink. This is the so called end-to-end security. The transport mode of IPSec is ideally suited for this purpose, and the communication endpoints are also IPSec endpoints. Depending on the policy, a single SA pair can secure all traffic between two endpoints (telnet, SMTP, Web, etc.), or unique SA pairs can protect each stream independently. In either case, traffic does not exist on the wire in an unsecured state (Fig. 3.2-13).

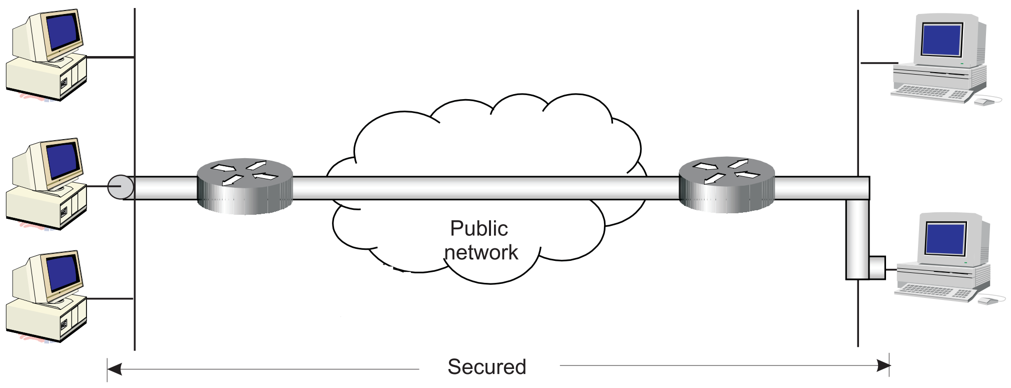


Fig. 3.2-13: End-to-End security through the network

Virtual Private Networks (VPNs)

VPNs with IPSec

A VPN can give a company the security it needs at a great savings in cost. It is not a physically distinct network (that's why it is ”virtual”), and tunnels are used to establish connectivity between disparate physical networks. A VPN is ”private” because tunnels are encrypted to provide confidentiality in a public network. In Fig. 3.2-14, the local networks LAN1 and LAN 2 are securely connected over the Internet. VPNs are constructed by deploying IPSec on routers that provide the physical network connection to a protected network. All traffic destined for the VPN is through SAs on the router and to a snooper in the network it looks the same - it is just encrypted packets sourced by one router and destined to another router. Since VPN are protecting transient traffic (traffic sourced or destined for hosts on a protected network), they must use IPSec in tunnel mode.

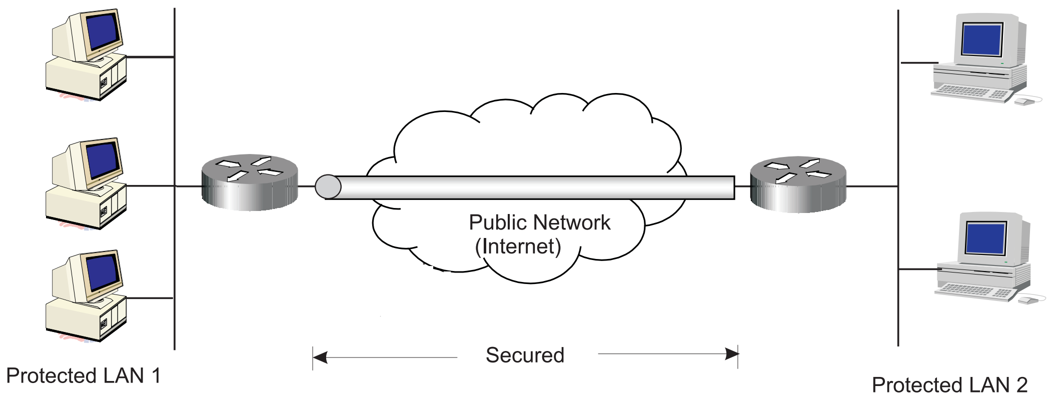


Fig. 3.2-14: A VPN across the Internet

Road Warriors

Road warriors with IPSec

In end-to-end security, packets are encrypted and decrypted by the hosts that produces and/or consumes the traffic. In a VPN, a router on the network encrypts and decrypts packets on behalf of a host on a protected network. A combination of the two is generally refered to as ”road warrior” configuration (Fig. 3.2-15). A road warrior is typically an individual who requires access to a protected network but does not reside at a fixed location (e.g. a treveling employee who must securely access corporate resources from a hotel, airport dial-up or from any Internet POP available). In this scenario, the road warrior's computer supports IPSec either natively, or by a bump-in-the-stack shim. He is able to secure outbound packets before they reach the wire and able to verify the security on inbound packets before they reach IP processing. The road warrior's IPSec peer is a router that protects the network he wishes to access.

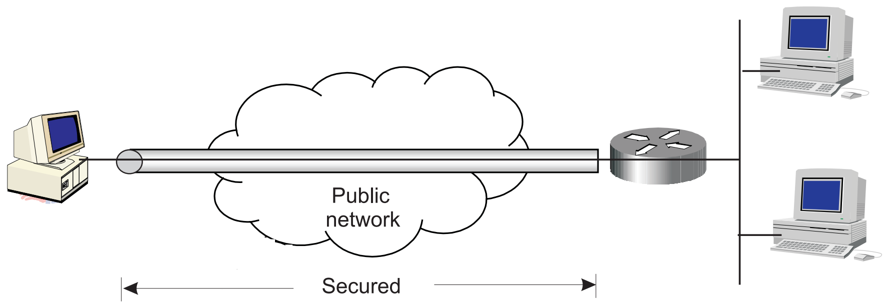


Fig. 3.2-15: Road warrior scenario

Nested Tunnels

Nested tunnels with IPSec

Look at the following example: A corporation has a security gateway to protect its network from competitors and hackers, but inside the network, some sensitive information (management information, acquisitions, personal data) has to be protected from its own employees. In this case multiple levels of network security need to be supported and nested tunnels are necessary (Fig. 3.2-16). In this case, the packets are secured two (or more times): once within the corporation, and then the outgoing traffic from the corporation is secured at the exit router once more. The packets are encapsulated at each point that requires additional security, and each encapsulation is atomic.

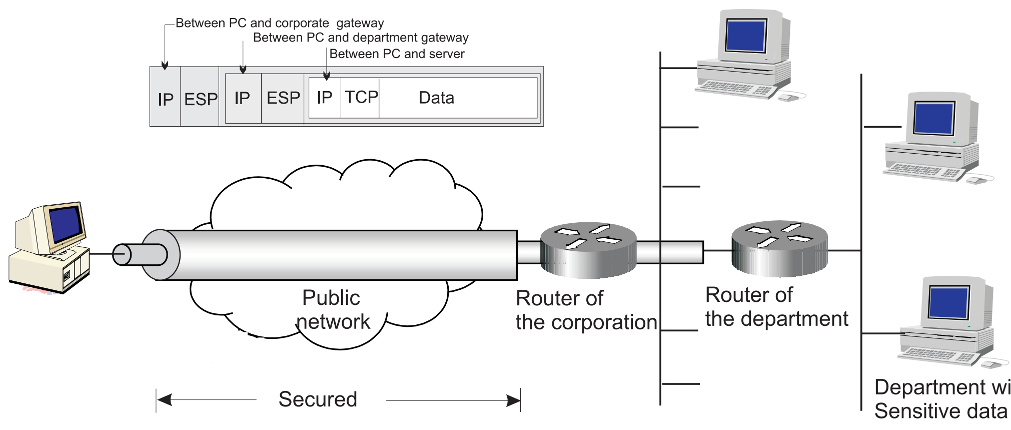

Between PC and corporate gateway
Between PC and department gateway
Between PC and server
IP
ESP
IP
ESP
IP
TCP
Data
Public network
Router of the corporation
Router of the department
Department with Sensitive data
Secured

Fig. 3.2-16: Nested tunnels

Chained Tunnels

Chained tunnels with IPSec

A common deployment of network security is a ”hub-and-spoke”. In such a design, there is usually a router that terminates multiple tunnels. Packets destined from one network to another are encrypted by the security gateway, decrypted by the hub, reencrypted, and decrypted by the security gateway protecting the remote network. Sometimes there are several hubs and the decryption and reencryption is done several times. Althought the performance of this design is not good as traditional VPN (the same packet is encrypted many times), this is a fairly common deployment.

The subjects of virtual private networks and firewalls which apply IPSec well be discussed in chapter 6 in more detail.

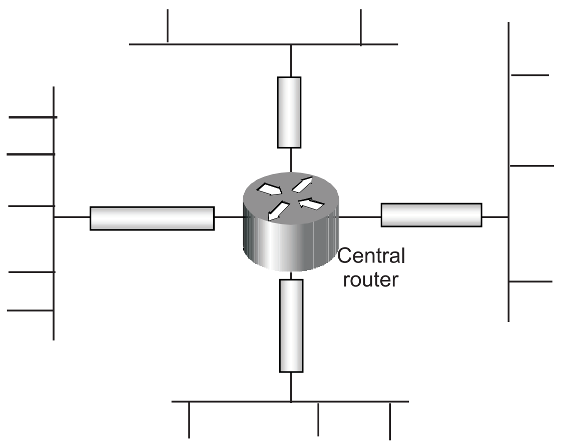


Fig. 3.2-17: Chained tunnels

3.2.11 Summary

The idea of providing security services at the network (i.e. IP) layer is not new, and several protocols have been proposed before IETF IPSec work group even started to meet. Examples include the security protocol 3 (SP3, developed by NSA and NIST), the network layer security protocol (NLSP, mainly developed by ISO), and the integrated NLSP (I-NLSP, developed by NIST). All these protocols use encapsulation as their enabling technique (authenticated and encrypted network layer packets are contained in other packets). But IPSec, designed from the IETF to provide cryptographically based security for IPv4 and IPv6 has become a standard.

The IP security architecture as described in this section is not an overall security architecture for the Internet. The topics of network address translation (NAT), support for IP multicast, and issues related to interoperability must be included in further studies. At present, most IPSec implementations are either Bump in the Stack (BITS) or Bump in the Wire (BITW). This is expected to change in future, since more and more vendors of networking software are integrating the suite of IPSec protocols into their products (including Cisco and Microsoft). Implementing the IPSec protocols is mandatory to IPv6 and therefore likely to become widespread during the next couple of years.

Exercise 3.2-1:

In discussing AH processing, it was mentioned that not all of the fields in an IP header are included in the HMAC calculation. For each of the fields in IPv4 header, indicate whether the field is immutable, mutable but predictable, or mutable (zeroed prior to HMAC calculation). Justify your answer for each field!

Exercise 3.2-2:

1. *Where may IPSec be implemented?*
2. *How can IPSec be implemented?*

3.3 Security at the Transport Layer

In this section we focus on the security protocols that have been proposed and implemented for the transport layer. The transport layer is responsible for providing services to the application layer. In the TCP/IP protocol suite the transport layer provides the following services:

- Connection-oriented or connectionless transport,
- Reliable or unreliable transport,
- Security (this service is new in comparison with the other services offered by the transport layer).

The TCP/IP protocol suite implements two protocols at the transport layer: TCP and UDP. The choice of using TCP or UDP is entirely up to the application - it has to choose the services it requires from the transport layer.

Transport layer security protocols

The idea of having a standardised **transport layer security protocols** is not new, and several protocols have been proposed before the IETF Transport Layer Security Working Group (IETF TLS WG) started to meet. The NSA[11] and NIST[12] developed the SP4 (Security Protocol 4) transport layer security protocol. The International Organisation for Standardisation (ISO) developed and standardised the Transport Layer Security Protocol (TLSP). In AT&T Bell Laboratories the Encrypted Session Manager (ESM) has been developed. We concentrate here on the most important protocols: Secure Shell (SSH, provides effective way to encrypt and authenticate network log sessions), the Secure Socket Layer (SSL) and the follow-on Internet standard of SSL known as Transport Layer Security (TLS). Since SSL is widely used and has become de-facto standard for protecting application's data in transit on the Internet, it will be described here in more detail.

3.3.1 Secure Shell (SSH)

Development

The Secure Shell (SSH) is a protocol and a software package that was originally developed by Tatu Ylönen at the Helsinki University of Technology (the later founder of SSH Communications Security). SSH was originally designed to replace the Berkeley r-tools such as **rlogin** (remote login), **rsh** (remote shell), **rcp** (remote file copy), and **rdist** (remote file distribution). Applying SSL they are reffered to as **slogin**, **ssh**, **scp**, and **sdist**. It can also replace Telnet in many cases. As such, SSH provides support for

- host and user authentication,
- data compression and
- data confidentiality and integrity protection.

Versions

There are currently two versions of SSH: a public and a commercial version. The public version[13] has been freely available for various UNIX systems since 1995. The commercial version is called F-Secure SSH, is available for many operating systems and equipped with a number of additional features and utilities. F-Secure SSH is used and widely deployed on the Internet today.

SSH and F-Secure SSH both utilize a generic transport layer security protocol. When used over TCP/IP, the server normally listens for TCP/IP connections on port 22, which has been registered for IANA[14] and is oficially assigned for SSH. More

11 National Security Agency (in USA)

12 National Institute of Standards and Technology (in USA)

13 Source and documentation are publicly available and can be downloaded from the SSH homepage on the WWW: http://www.cs.hut.fi/ssh

14 The IANA (Internet Assigned Name Authority) is the overall authority for day-to-day administration of the Internet Domain Name System (DNS). IANA staff carry out administrative responsibilities for the assignment of IP Addresses, Autonomous System Numbers, Top Level Domains (TLDs), ports, and other unique parameters of the DNS and its protocols.

recently, IETF has chartered Secure ShellWorking Group (secsh WG) to standardize version 2 of th SSH protocol[15].

Subprotocols of SSH

The SSH protocol consists of three major components: The SSL Transport Layer Protocol provides server authentication, confidentiality, and integrity with perfect forward secrecy. The SSL User Authentication Protocol authenticates the client to the server. The SSL Connection Protocol multiplexes the encrypted tunnel into several logical channels. Each protocol is specified in an Internet Draft. Here, we only consider the most important parts of this protocols.

SSH Transport Layer Protocol

The SSH Transport Layer Protocol provides cryptographic host authentication, data confidentiality and data integrity. The protocol does not provide user authentication (which is, as mentioned, provided by the SSH Authentication Protocol). The SSH Transport Layer Protocol supports various key exchange, encryption, and MAC algorithms that are negotiated during the connection establishement. Some algorithms are mandatory to implement (all implementations are required to support them), other algorithms are defined in the protocol specification but are optional.

Protocol start

When the SSH protocol is used to establish a TCP/IP connection between a client and a server, the two parties first exchange identification strings that contain the SSL protocol and software version numbers in use. When a protocol execution starts, no encryption and message authentication are in use, but after algorithms negotiation and key exchange, compression, encryption and authentication are applied to all subsequent messages.

Message flow

The **message flow**s on the SSH transport layer protocol executed between a client C and a server S is shown in Fig. 3.3-1 (simplified) and can be summarized as follows:

15 http://www.ietf.org/html.charters/secsh-charter.html

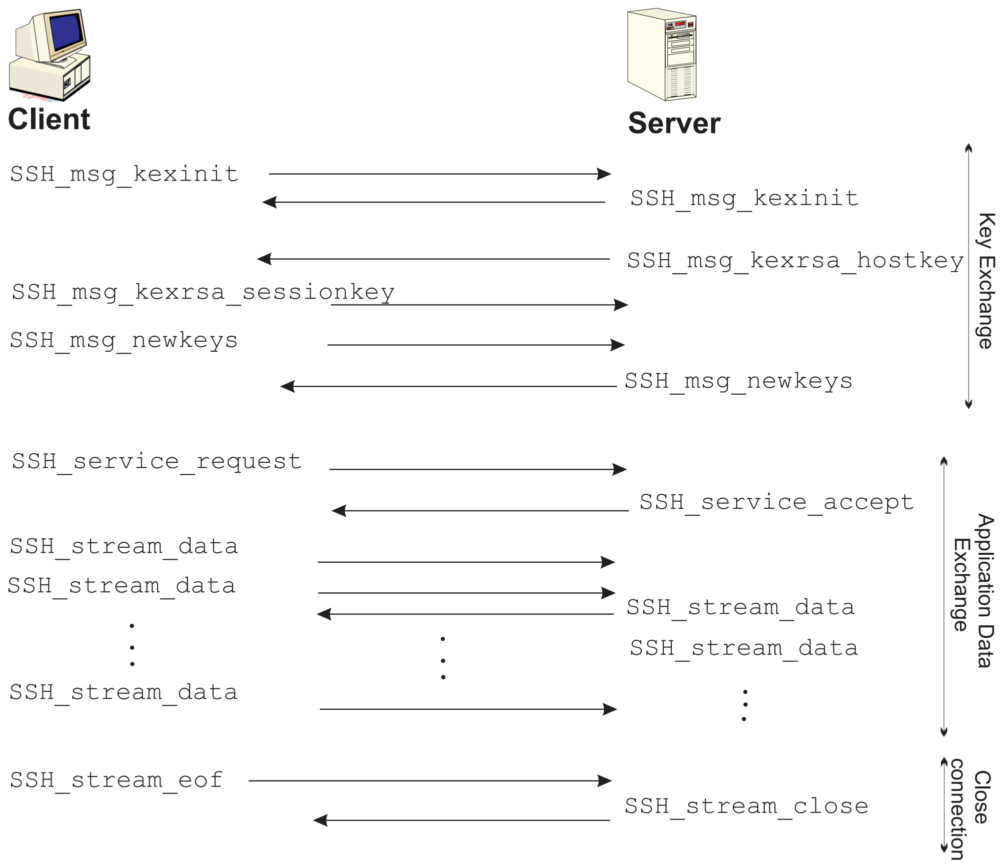


Fig. 3.3-1: Simplified message flow in SSH transport protocol

Algorithms negotiation

The **first phase** after the protocol start is algorithm negotiation. The client sends a Key Exchange Initialisation (`SSH_msg_kexinit`) message to the server. This message includes a list of supported data compression, encryption, MAC, and key exchange algorithms. The first algorithm in each list is the prefered algorithm. For each algorithm type, the server selects the first algorithm of the client's list, which it also supports. The list of its own algorithm preferences, the server sends to the client with the server Key Exchange Initialisation (`SSH_msg_kexinit`) message. So, in this step, the client and the server agreed upon the algorithms they will use in the following session. Currently, the following algorithms are defined in SSH:

- Data encryption algorithms: Triple DES in CPC mode (mandatory), IDEA in CBC mode, ARCFOUR stream cipher, Blowfish in CBC mode.
- Message authentication algorithms: HMAC-MD5, HMAC-SHA, SHA of key+data+key (mandatory), MD5-8 (first 8 bytes of MD5 of key+data+key), and SHA-8 (first 8 bytes of SHA5 of key+data+key).

Key exchange

After the `kexinit` packet exchange, the **key exchange** algorithm is run. The key exchange produces two values: a shared secret *K* and the exchange hash *H*. Encryption and authentication keys are derived from these. The server starts the key exchange by sending an `SSH_msg_kexrsa_hostkey` message to the client, which include its public key. When the client receives this message, it first verifies whether the server key belongs to the intended server. Then it generates a 256-bit random secret *K*. In order to be sure that no one has manipulated the key exchange so far, the client computes a hash value *H* from the concatenated payloads of the client and server `SSH_msg_kexinit` messages, the server `SSH_msg_kexrsa_hostkey` message, and the session key. This value is called **session identifier** (`session id`) and is used to to authenticate the key exchange as a whole. Then the client concatenates six 0 bytes, the first 10 bytes of the session identifier and the 32 bytes of the shared secret *K*, and encrypts the resulting 48 bytes with the server's public key. The corresponding `SSH_msg_kexrsa_sessionkey` message is sent to the server. After receiving this message, the server uses its private key to decrypt the secret *K*. As a result of the key exchange, the client and server share a 256 bit secret *K* that can now be used to compute session keys[16]. The key exchange ends by each side by sending an SSH msg newkeys message. All messages sent after this message must use the new keys and algorithms.

Key reexchange

Either side may request a **key reexchange** at any time after the initial key exchange by sending a new `SSH_msg_kexinit` message to the other. All keys and initialisation vectors are recomputed after the reexchange. The SSH protocol specification recommends performing a key reexchange after each gigabyte of transfered data or after one hour of connection time.

Service request

After the key exchange, the client requests a service (e.g. executing a command on the server, file transfer). It does it by sending the message `SSH_service_request` to the server. If the server supports the service and permits the client to use it, it responds with a corresponding `SSH_service_accept` message. Each service is identified by a unique service name. The client must wait for response to its service request message before sending any further data.

Data transmission

Once a particular service has been selected, data are transmitted with `SSH_stream_data` messages. The data fields of the message are encrypted with the selected algorithm and key. The message also contains the MAC of the data computed with the selected MAC algorithm and key. The encryption steps in each direction (client->server or server->client) run independently and they typically use different keys. Different encryption algorithms may also be used in each direction. Note that Triple DES is the only encryption algorithm that is mandatory for SSH 2.0. The MAC is computed over the actual content of the message and a 32-bit sequence number (The sequence number of the first message is set to zero; then it is incremented by one for each packet sent.). The MAC is

16 The session keys are computed as follows:

- - encryption key client to server: $Hash(K||H||''C''||session\_id)$
- - encryption key server to client: $Hash(K||H||''D''||session\_id)$
- - MAC key client to server: $Hash(K||H||''E''||session\_id)$
- - MAC key server to client: $Hash(K||H||''F''||session\_id)$

The letters "A", "B", "C" and "D" are encoded as ASCII characters. Key data must be taken from the beginning of the hash output.

transmitted without encryption as the last part of the SSH message. The applied MAC algorithms and keys can also be different for each direction.

Closing the connection

When the server or client application closes its output, a `SSH_stream_eof` message is sent to the other side. When either side wants to terminate the communication, it sends a `SSH_stream_close` message. The session ends by sending back a `SSH_stream_close` message from the other side.

Either party may send disconnect, ignore, debug or other reserved messages at any time.

SSH Authentication Protocol

Purpose

The **purpose** of SSH authentication protocol is to perform client user authentication. It runs on top of the SSH transport layer protocol, which has already authenticated the server machine, established an encrypted communication channel, and computed a unique session identifier for this session. The SSH authentication protocol provides a single authenticated tunnel for the SSH connection protocol.

Protocol

When a client declares its user name and the service name it wants to use, user authentication is required. The server drives the authentication by telling the client which authentication methods can be used. The client has the freedom to try the methods listed by the server in any order. The server should have a timeout for authentication, and disconnect if the authentication has not been accepted within this period. Additionaly, the implementation should limit the number of failed authentication attempts. The dialogue continues untill access has been granted or denied.

Supported user authentication methods

Several user authentication methods are predefined in the SSH protocol specification. This include password authentication, SecurID authentication, S/Key and OPIE one-time password authentication, authentication with public key, host-based or Kerberos authentication. Additional authentication methods can also be defined.

In the case of the password method, the authenticating informtion is basically a user password. The password is sent in clear text, but the entire SSH message (including the password) is encrypted by the SSH transport layer protocol. Most one-time password methods use a challenge-response authentication: a server sends a challenge to the client and the client returns an appropriate response[17]. The public key and host based authentication methods work by having the client send a signature with a private key. In the several possible variants of Kerberos authentication, the messages typically include some sort of user credentials or tickets.

SSH Connection Protocol

Channel mechanism

All terminal sessions, forwarded connections, etc. are channels. Either side may

17 For more information see [KCL+00].

open a channel. Multiple channels are multiplexed into a single connection. Channels are identified by numbers at each end. The number referring to a channel may be different on each side. Requests to open a channel contain the sender's channel number. Any other channel-related messages contain the recipient's channel number for the channel.

Purpose The SSH Connection Protocol provides interactive login sessions, remote execution of commands, forwarded TCP/IP connections, and forwarded X11 connections. All of these **channels are multiplexed into a single encrypted tunnel**. The SSH Connection Protocol has been designed to run on top of the SSH transport layer and user authentication protocols. Here we will not address the SSL Connection Protocol in detail. The interested reader can find more information in the SSH specification.

Summary

The Secure Shell protocol is the de-facto standard for encrypted and authenticated **remote terminal connections** and secure file transfers on the Internet. It is a full replacement for telnet, rlogin, rsh, rcp, and ftp.

This protocol ia designed to be used over a reliable transport. If transmission errors or message manipulations occur, the connection is closed. Disadvantge of SSH is that it uses manually distributed and preconfigured public keys instead of certificate-based key management. This is notably the major disadvantage of SSH compared to SSL and TLS protocols.

Exercise 3.3-1:

*Answer **briefly** the following questions:*

1. *What is F-Secure SSH?*
2. *On which port listen server applications using SSH?*
3. *Which are the main components (subprotocols) of the SSH protocol?*
4. *What is the goal of the SSH transport layer protocol?*
5. *Which data encryption algorithms can be used in SSH?*
6. *The SSH protocol specification recommends performing a key reexchange after each gigabyte transfered data or after one hour of connection time. Why does this key reexchange make sense?*
7. *Can the SSH protected data in a connection be differently encrypted and/or authenticated in each direction?*

3.3.2 Secure Socket Layer (SSL)

The application layer defines the interface to the transport layer, which is operating system dependent. The most popular interface is the **socket** interface. The socket interface is provided in all flavours of the UNIX operating system and on the Microsoft platforms. The SSL protocol is based on the concept of socket.

Development

Netscape Communications introduced the SSL method and a corresponding protocol with the first version of Netscape Navigator. Netscape Communications did not charge its customers for the implementation of its security protocol and consequently, SSL became after 1994 the predominant protocol to provide security services on the transport layer. There have been three versions of SSL: The first version, SSL 1.0, contained some serious flaws and was never released to public. The second version, SSL 2.0, was incorporated into Netscape Navigator versions 1.0 through 2.x. and it had some security weaknesses, to which Netscape answered with the version SS v3.0, which also added some new features. The latest specification of SSL 3.0 was oficially released in March 1996. It is implemented in both Netscape Navigator 3.0 (and higher) and Microsoft Internet Explorer[18] 3.0 and higher. SSL has become a de facto standard for cryptographic protection of HTTP traffic.

Goals of SSL

The primary goal of the SSL Protocol is to provide cryptographic security between two communicating applications. A second goal of SSL is extensibility: It provides the framework into which new public key and bulk encryption methods can be incorporated as necessary. SSL also tends to achieve interoperability between different independent applications which use SSL and relative efficiency of the cryptographic operation. It provides **connection security** with the following properties:

- The connection is private. In the initial handshake protocol secret keys are defined, then symmetric cryptography is used for data encryption.
- The peer's identity can be authenticated using asymmetric cryptography.
- The connection is reliable. Message transport includes a message integrity check using a keyed MAC.

In order to make use of SSL, the client and server must both know that the other side is using SSL. One (and the common) possibility to do this is to use dedicated port numbers reserved by the IANA. In this case, a separate port number must be assigned for every application protocol with SSL support. For example, https (http over SSL/TLS) uses port 443, ftps (ftp over SSL/TLS) uses port 990, telnets (telnet over SSL/TLS) uses port 992, smtps[19] (smtp over SSL/TLS) uses port 465, and so

18 Microsoft has attempted to leverage public uncertainity about SSL's security and introduced the competing `Private Communicating Technology` (PCT) protocol in its first release of Internet Explorer 1996. But after the release of SSL 3.0 Microsoft backed down and agreed to support SSL in all versions of its TCP/IP-based software.

19 The "s" on the end of the protocol name indicates that the SSL is consistentely used for securing the protocol.

on. Another possibility is to use the normal port number for every application and to negotiate security options as part of the modified application protocol.

The following description of SSL is based on the latest protocol specification of SSL 3.0.

SSL Architecture

SSL in the TCP/IP stack

SSL is designed to make use of TCP to provide a reliable end-to-end secure service. It operates on the top of the transport layer, i.e. between the transport and the application layer, as shown in Fig. 3.3-2. Thus, it can protect the application data and the header from the application layer, but the control data from the lower layers (transport layer header, IP layer header and network access layer header) are transmitted unprotected (Fig. 3.3-3). The SSL protocol provides security services to various higher layer protocols. For example, the hypertext transfer protocol (HTTP), which provides the transfer service for Web client-server interaction, can operate on top of SSL.

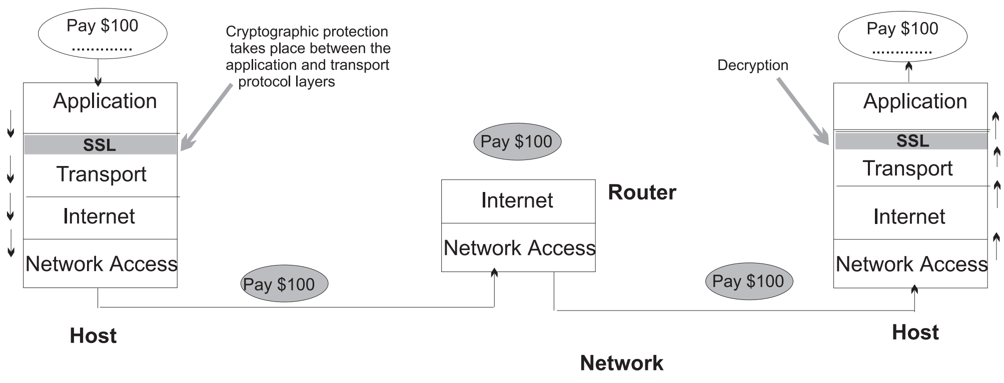


Fig. 3.3-2: SSL in TCP/IP stack

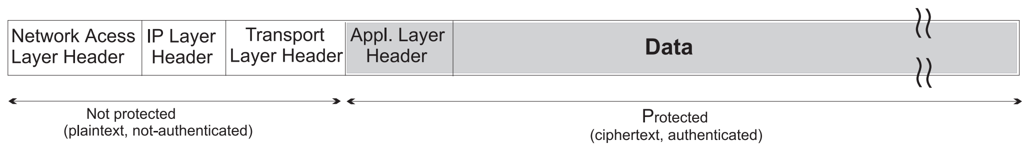

Network Acess Layer Header
IP Layer Header
Transport Layer Header
Appl. Layer Header
Data
Not protected (plaintext, not-authenticated)
Protected (ciphertext, authenticated)

Fig. 3.3-3: SSL packet format

It is important to note that the SSL protocol does not protect against traffic analysis attacks. For example, by examinning the unencrypted IP source and destination adresses and the TCP port numbers, or examining the volume of network traffic flow, a traffic analyst can eventually determine what parties are interacting, what types of services are being used, or sometimes recover information about business or personal relationships.

SSL is not a single protocol but rather two layers of protocols, as illustrated in Fig. 3.3-4. In the higher layer of SSL operate three protocols: the Handshake Protocol, the Change Cipher Spec Protocol, and the Alert Protocol. These SSL-specific protocols deal with the SSL management and will be explained later. In the lower layer of SSL operates the SSL Record Protocol which in fact provides the security services (encryption and authentication of data).

Layers of SSL

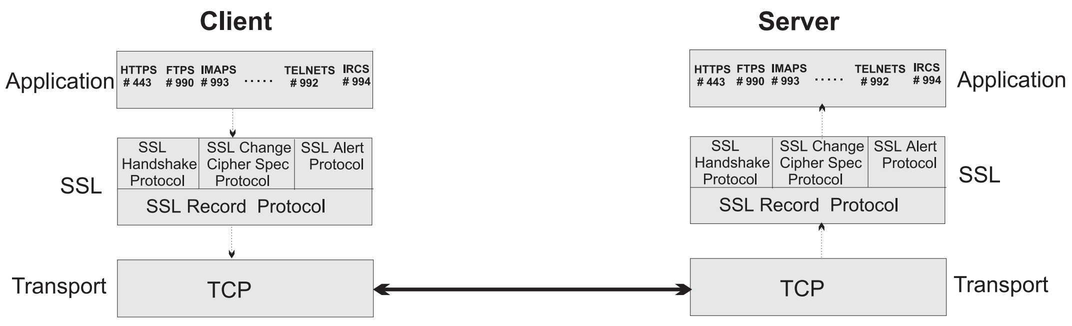


Fig. 3.3-4: SSL architecture

Two important SSL concepts are the **SSL connection and SSL session**. A **connection** is a transport that provides a suitable type of service. For SSL such connections are peer-to-peer relationships, they are transient and every connection is associated with one session. An **SSL session** is an association between a client and a server. Sessions are created by the Handshake Protocol and they define a set of cryptographic security parameters (algorithms, keys) which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection. There are a number of **states** associated with each session. Logically the state is represented twice, once as the current **operating state**, and (during the handshake protocol) again as the **pending state**. Once a session is established, there is a current operating state for both read and write (i.e. receive and send). Actually, during the Handshake Protocol pending read and write states are created which become current states after the successful conclusion of the Handshake Protocol.

SSL connection and SSL session

A **session state** is defined with the following parameters: session identifier, peer certificate, compression method, cipher spec, master secret, and the "is resumable" field. The significance of these parameters is explained in Fig. 3.3-5.

Session state

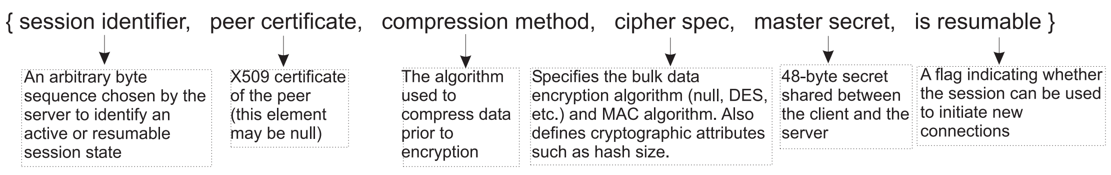
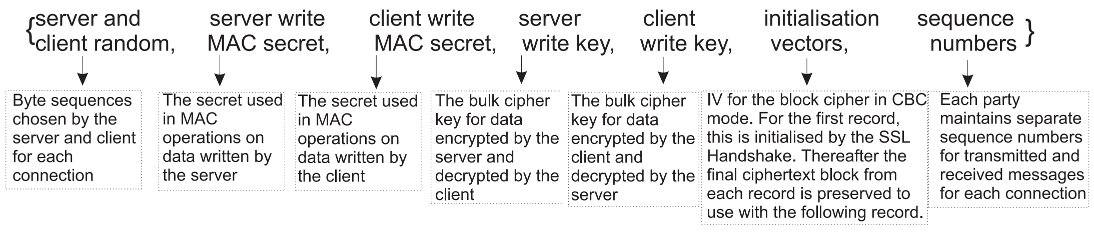

{ session identifier, peer certificate, compression method, cipher spec, master secret, is resumable }
An arbitrary byte sequence chosen by the server to identify an active or resumable session state
X509 certificate of the peer (this element may be null)
The algorithm used to compress data prior to encryption
Specifies the bulk data encryption algorithm (null, DES, etc.) and MAC algorithm. Also defines cryptographic attributes such as hash size.
48-byte secret shared between the client and the server
A flag indicating whether the session can be used to initiate new connections

Fig. 3.3-5: SSL session state

Connection state

A **connection state** is defined with the following parameters: server and client random, server write MAC secret, client write MAC secret, server write key, client write key, initialization vectors, and sequence numbers. The significance of these parameters is explained in Fig. 3.3-6.



Byte sequences chosen by the server and client for each connection

The secret used in MAC operations on data written by the server

The secret used in MAC operations on data written by the client

The bulk cipher key for data encrypted by the server and decrypted by the client

The bulk cipher key for data encrypted by the client and decrypted by the server

IV for the block cipher in CBC mode. For the first record, this is initialised by the SSL Handshake. Thereafter the final ciphertext block from each record is preserved to use with the following record.

Each party maintains separate sequence numbers for transmitted and received messages for each connection

Fig. 3.3-6: SSL connection state

It is the responsibility of the SSL Handshake protocol to coordinate the states of the client and the server, thereby allowing the protocol state machine of each to operate consistently. The SSL begins with the Handshake Protocol, but because it is most complex part of SSL, we first consider the another parts of SSL, beginning with the SSL record layer.

SSL Record Protocol

Goal of the protocol

The SSL Handshake protocol defines a shared secret key that is used for conventional encryption of SSL payloads and a shared secret key that is used to form a MAC. After that, the SSL record layer receives uninterpreted data from higher layers in non-empty blocks of arbitrary size and provides security services to this data, i.e the SSL Record Protocol provides confidentiality and message integrity for the SSL connection.

Fragmentation

The steps of the SSL Record Protocol are illustrated in Fig. 3.3-7. The first step is **fragmentation**. The record layer takes an application message to be transmitted and fragments this information block into SSL plaintext records of 2^{14} bytes or less. This data is transparent and treated as an independent block to be dealt with by the higher level protocol.

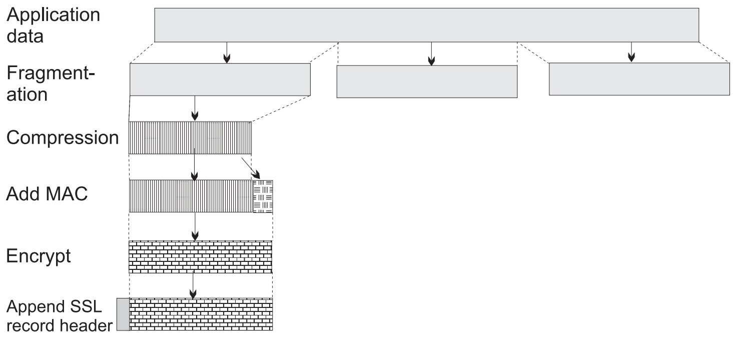


Fig. 3.3-7: SSL record protocol

Compression

Next, **compression** is optionally applied. All records are compressed using the compression algorithm defined in the current session state. The compression algorithm translates an SSL Plaintext structure into an SSL compressed structure. The next step in processing is to compute a **Message Authentication Code** (MAC) over the compressed data. The MAC algorithm to be used is defined in the current CipherSpec (keyed MD5 or keyed SHA-1) and the shared secret key has been assigned in the Handshake Protocol.

Message Authentication

Encryption

After the MAC computation, the compressed message plus the MAC are **encrypted** using symmetric **iptables**. Just as in the case of MAC, the encryption algorithm to be used is defined in the current CipherSpec and the shared secret key has to be agreed in the Handshake Protocol. The following block encryption algorithms are permitted: IDEA (128 bit key), RC2-40 (40 bit key), DES-40 (40 bit key), DES (56 bit key), 3DES (168 bit key), and Fortezza (80 bit key). Moreover, the stream ciphers RC4-40 (40 bit key) and RC4-128 (128 bit key) can be used. For the CBC mode, the initialisation vector (IV) for the first record is provided by the handshake protocol. The IV for subsequent records is the last ciphertext block from the previous record. Note that the MAC is computed before encryption. The encryption and MAC functions translate an SSL Compressed structure into an SSL Ciphertext.

SSL record header

The final step of SSL Record Protocol processing is to **prepend a header**. It consist of four fields: Content Type (8 bits), Major Version (8 bits), Minor Version (8 bits) and Compressed Length (16 bits). The Content Type field indicates the higher-layer protocol used to process the enclosed fragment, i.e. it can be `change_cipher_spec`, `alert`, `handshake`, and `application_data` (see Fig. 3.3-8). Note that no distinction is made among various applications that might use SSL (e.g. HTTP or FTP). The fields Major Version and Minor Version indicate the major resp. the minor version of SSL used. For example, for SSLv3, the Major Version value is 3 and the Minor is 0. The field Compressed Length contents the length in bytes of the plaintext fragment, or compressed fragment (if compression is used).

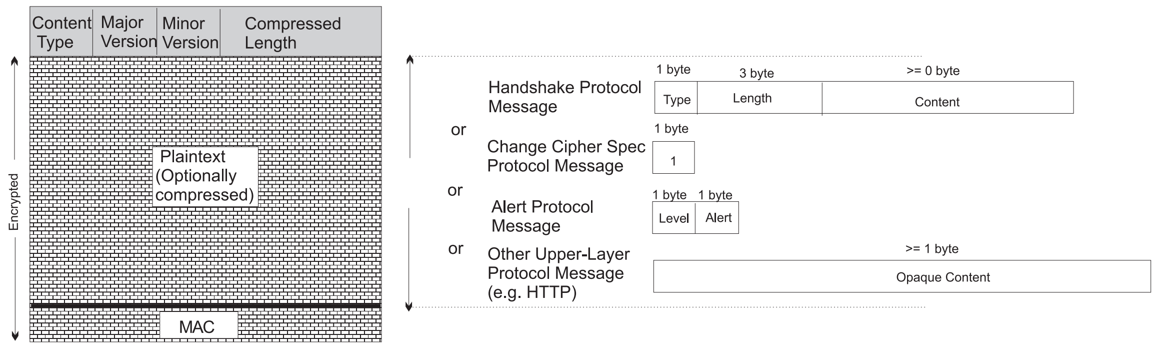


Fig. 3.3-8: SSL record format

At the receiving site, the operations in the SSL Record Layer are performed vice versa: first the SSL record layer header is removed, then the record is decrypted, the MAC is verified and the data is decompressed.

Change Cipher Spec Protocol

The Change Cipher Spec Protocol is one of the three SSL specific protocols that use the SSL Record Protocol (see Fig. 3.3-4), and it is the simplest. It signals transitions in ciphering strategies. The protocol consists of a single message, which consists of a single byte with value 1 (see Fig. 3.3-8). This message is sent by both client and server to notify the receiving party that subsequent records will be protected under the just-negotiated CipherSpec and keys. Reception of this message causes the receiver to copy the read pending state into the read current state, i.e. the purpose of this message is to cause the pending state to be copied into the current state, which updates the cipher suite to be used in the connection. Change Cipher Spec Messages are sent during the Handshake Protocol following the handshake key exchange and certificate verify messages (see Fig. 3.3-9 and Fig. 3.3-10).

Goal of the protocol

Alert Protocol

One of the four content types supported by the SSL record layer is the alert type. The Alert Protocol is used to convey SSL-related alerts to the peer entity. Alert messages convey the severity of the message and a description of the termination of the communication. In this case, other connections corresponding to the session may continue, but the session identifier must be invalidated, preventing the failed session from being used to establish new connections. Like other messages, alert messages are encrypted and compressed, as specified by the current connection state.

Goal of the protocol

Closure and error alerts

Two important types of alerts are closure and error alerts. The client and the server must share knowledge that the connection is ending and either party may initiate the exchange of closing messages, i.e. closure alerts. The error alerts signalise the occurance of an error. Error handling is in the SSL handshake protocol very simple. When an error is detected, the detecting party sends a message to the other party. Upon transmission or receipt of a fatal alert message, both parties immediately close the connection. Servers and clients are required to forget any session-identifiers, keys, and secrets associated with the failed connection.

Alert protocol message

Each message in the alert protocol consists of two bytes (Fig. 3.3-8). The first byte takes the value warning (1) or fatal (2). The second byte contains a code that indicates the specific alert. Examples of defined error alerts in SSL are: `unexpected_message` (an inappropriate mesage was received), `bad_record_mac` (a record is received with an incorrect MAC), `decompression_failure` (unable to decompress), `handshake_failure` (the sender was unable to negotiate an acceptable set of security parameters), `no_certificate`, `bad_certificate`, `unsupported_certificate`, `certificate_revoked`, `certificate_expired`, `certificate_unknown` and `illegal_parameter`.

Handshake Protocol

Goal of the protocol

The cryptographic parameters of the session state are negotiated by the SSL Handshake Protocol, which operates on top of the SSL record layer. The handshake is performed before any application data is transmitted. When an SSL client and server first start communicating, they agree on a protocol version, select cryptographic algorithms, optionally authenticate each other, and use public-key encryption techniques to generate shared secrets. These processes are performed in the Handshake Protocol, which we explain in the following.

Handshake Protocol message

The Handshake Protocol consists of a series of messages exchanged by client and server. Each message consist of three fields: Type, Length, and Content (see Fig. 3.3-8). The field **Type** is one byte and indicates one of 10 message types in the protocol. They are `hello_request`, `client_hello`, `server_hello`, `certificate`, `server_key_exchange`, `certificate_request`, `server_done`, `certificate_verify`, `client_key_exchange` and `finished`. The field **Length** (3 bytes) indicates the length of the message in bytes. The field **Content** (≥ 1 byte) comprises the parameters associated with this message.

First phase of the handshake

The handshake protocol can be viewed as having four phases. In the **first phase security capabilities are established**. The phase is initiated by the client and is used to initiate a logical connection by exchanging a `Hello` messages between the client and the server.

When a client first connects to a server it is required to send the `Client Hello` as its first message (see Fig. 3.3-9). The client can also send a `Client Hello` in response to a `Hello Request` message that comes from the server or on its own initiative in order to renegotiate the security parameters in an existing connection. The `Client Hello` message is sent with the following parameter: `Version` (the highest SSL version understood by the client, `Session ID` (session identifier defined by the server and used to identify a session between client and server), `Random Structure` (28 random bytes generated by a secure random number generator and used later in the protocol), `Cipher Suite`[20] (list which contains the combinations of cryptographic algorithms supported by the client), and `Compression Method` (a list of compression algorithms supported by the client; if the server supports none of those specified by the client, the session must fail).

20 Each Cipher Suite defines both a key exchange algorithm and a CipherSpec. The first element of the Cipher Suite is the key exchange method and the second is CipherSpec. The following key exchange methods are supported: RSA, authenticated Diffie-Hellman, anonymous Diffie-Hellman, and Fortezza. The CipherSpec includes the following fields: cipher algorithm (RC4, RC2, DES, 3DES, DES40, IDEA, Fortezza), MAC algorithm (MD5 or SHA-1), cipher type (stream or block), hash size (0, 16 or 20 bytes), key material (a sequence of bytes that contain data used in generating the keys), and IV size (size of the initialisation vector for CBC mode).

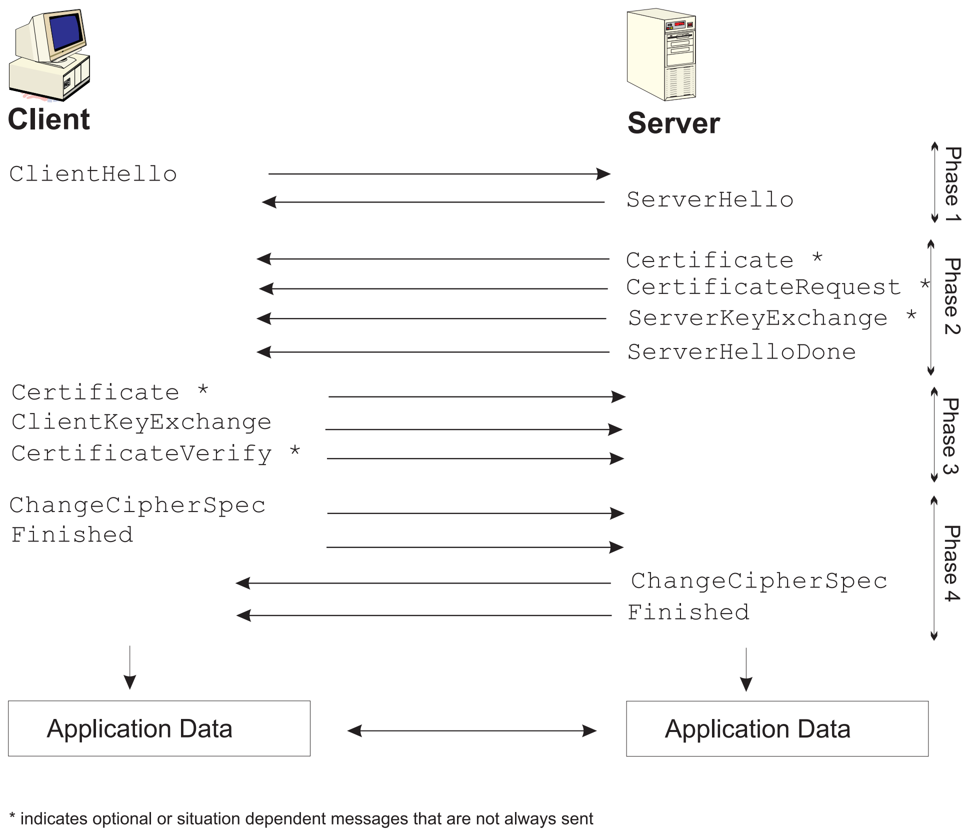


Fig. 3.3-9: SSL handshake protocol

The server processes the `Client Hello` message and responds with either a `Handshake Failure` (fatal error will occur and the connection will fail) or a `Server Hello` message. The `Server Hello` message contains the same parameters as the `Client Hello` message: `Version`, `Random`, `Session ID`, `Cipher Suite`, and `Compression Method`. The `Version` field contains the lowest of the version suggested by the client and the highest supported by the server, and the `Random` field is generated by the server. If the server is willing to establish the new connection using the specified session state, it will respond with the same value of session identifier supplied by the client. The `Cipher Suite` parameter contains a single cipher suite selected by the server from the list of cipher suites proposed by the client (if no acceptable choices are presented, the server returns a `Handshake Failure` alert and closes the connection). The `Compression Method` is selected by the server from the list of compression algorithms in the `Client Hello` message.

So, the `Client Hello` and `Server Hello` messages are used to establish security enhancement capabilities between client and server, i.e. the protocol version, session ID, cipher suite, and compression method are established.

Second phase of the handshake

The **second phase** of the handshake protocol is the **Authentication and Key Exchange**. If the server is to be authenticated (which is generally the case), it sends its certificate in a `Certificate` message immediately following the `Server Hello` message. The certificate type must be appropriate for the selected cipher suite's key exchange algorithm, and is generally an X.509.v3 certificate.

If the server has no certificate, or has certificate only used for signing (e.g. DSS certificates, signing only RSA certificates), or Fortezza/DMS key exchange is used, the server sends the `Server Key Exchange` message. With this message, the server informs the client about its public key parameters i.e. the message contains the parameters used for RSA key exchange (RSA modulus and the public exponent of the server's temporary RSA key), the parameters used for Diffie-Hellman key exchange (prime modulus p, generator g, and the server's public value $g^x \bmod p$), or the parameters for the Fortezza key exchange algorithm.

When the server is authenticated, it may request a certificate from the client if that is appropriate to the cipher suite selected, and send the `Certificate Request` message. This message contains the types of certificates requested and a list of names of acceptable certificate authorities. It is a fatal handshake failure alert for anonymous server to request client identification.

The final message in phase 2, and the one that is always required, is the `Server Hello Done` message sent by the server. This indicates that the `Server Hello` and associated messages phase of the handshake is complete. The server will then wait for a client response - the client should verify that the server provided a valid certificate (if required) and that the `Server Hello` parameters are acceptable.

Third phase of the handshake

The **third phase** of the Handshake is the **client authentication and key exchange**. Upon receipt of `Server Hello Done` message, the client should verify that the server provided a valid certificate if required and check that the `Server Hello` parameters are acceptable. If all is satisfactory, the client sends one or more messages back to the server. If the server has sent a `Certificate Request` message, the first message the client sends is its `Certificate` message (or a `No Certificate` alert, if no suitable certificate is available). This message is the same type as the server's `Certificate` message.

Next is the `Client Key Exchange` message, and the content of this message depends on the type of key exchange selected between the `Client Hello` and the `Server Hello`[21]. Finally, in this phase, the client may send a `Certificate Verify` message to provide explicit verification of the client certificate (if the client has sent a certificate). This message signs a hash code based on the preceeding messages in the handshake.

21 If RSA is being used for key agreement and authentication, the client generates a 48-byte premaster secret, encrypts it under the public key from the server's certificate or temporary RSA key from the Server Key Exchange message, and sends the result in an Encrypted Premaster Secret message. This random value (pre-master-secret) is generated by the client and is used to generate the master secret. If Fortezza key exchange is being used, the client derives a Token Encryption Key (TEK) using the Fortezza algorithm. This calculation uses the public key in the servers certificate along with private parameters in the clients token. The client sends public parameters needed for the server to generate the TEK and generates session keys, wraps them using the TEK and sends the result to the server. If the Diffie-Hellman key exchange algorithm is being used, the Client Key Exchange message conveys the client's Diffie-Hellman public value, if it was not already included in the client's certificate.

Fourth phase of the handshake

The **fourth phase**, i.e. the **finish phase** completes the setting up of a secure connection. The client sends a `Change Cipher Spec` message and copies the pending `Cipher Spec` into the current `Cipher Spec`. Note that this message is not considered part of the Handshake Protocol but is sent using the Change Cipher Spec Protocol. The client then immediately sends the `Finished` message to verify that the key exchange and authentication process were successful. This is the first message protected with the negotiated algorithms and keys. This message includes hash values of all handshake messages starting at `Client Hello` up to the `Finished` message.

In response of these two messages, the server sends its own `Change Cipher Spec` message, transfers the pending to the current `Cipher Spec`, and then sends its own `Finished` message under the new Cipher Spec. Recipients of `Finished` message must verify that the contents are correct. No acknowledgement of the `Finished` message is required.

End of the handshake

At this point, the handshake is complete and the client and server may begin to exchange application layer data. The application data messages are carried by the record layer and are fragmented, compressed and encrypted based on the current connection state. The handshake protocol messages are presented in the order they must be sent; sending handshake messages in an unexpected order results in a fatal error.

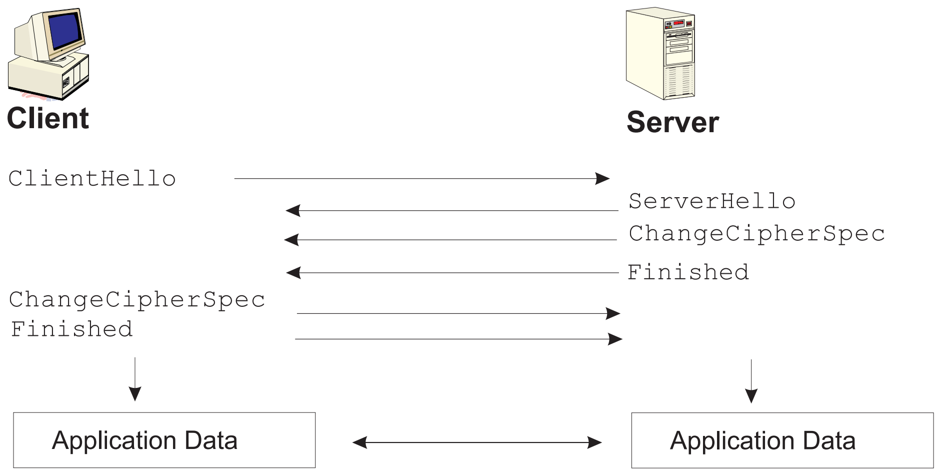


Fig. 3.3-10: SSL handshake protocol when resuming or duplicating the previous session

Resuming a privious or dublicating an existing session

When the client and server decide to **resume a previous session or duplicate an existing session** (instead of negotiating new security parameters) the message flow is as follows (Fig. 3.3-10). The client sends a `Client Hello` message using the session ID of the session to be resumed. The server then checks its session cache for a match. If a match is found, and the server is willing to reestablish the connection under the specified session state, it will send a `Server Hello` with the same Session ID value. At this point, both client and server must send `Change Cipher Spec` messages and proceed directly to finished messages. Once the reestablishment is complete, the client and server may begin to exchange application layer data. If a Session ID match is not found, the server generates a new session ID and the SSL client and server perform a full handshake.

SSL Implementation

With version 3.0, SSL has become a fairly sophisticated protocol. It provides its security services without requiring users to manipulate cryptographic keys personally. For example, if a hypertext link in a page specifies ”https”, then the browser would automatically try to use SSL without requiring extra work by the user. In the recent past, many Web servers and browsers have been modified to make use of SSL. There are several commercial Web servers available that support SSL (and TLS). On the client side, most browsers (e.g. Netscape Navigator, Internet Explorer) provide support for SSL. The three major implementations of SSL (SSLref, SSLeay, and SSL Plus) will be briefly discussed below. In addition, there is an increasingly large number of SSL implementations based on the Java programming language.

SSLref **SSLeay** **SSL Plus**

Netscape has developed an SSL reference implementation of the Version 3.0 specification called **SSLref**. The implementation was written in C programming language and published in 1995. The SSLref source code is distributed freely for noncommercial use. Parties interested in using SSLref in a commercial product require a licence from Netscape Communications. Some of algorithms used by SSL (RC2, RC4, RSA) must be licenced. **SSLeay** is an independent implementation of SSL, developed by Eric Young in Australia. SSLeay implements both SSL 2.0 and SSL 3.0 and TLS 1.0. The software is publicly and freely available around the world on a number of anonymous FTP sites. Since SSLeay was entirely developed outside the USA, its distribution is not restricted by U.S. export controls. The **SSL Plus** Security Toolkit developed by Consensus Development Corporation is currently the leading commercial implementation of SSL 3.0. It is licenced and used by several leading companies to deploy security using public key cryptography and digital certificates for their applications.

For obvious reasons, the use of SSL slows down the speed of a browser interacting with an HTTPS server. To minimize the performance degradation and impact of SSL, some organisations transmit the bulk of their data in clear text, and use SSL only for encrypting sensitive data. Unfortunately, this leaves the user open to specific attacks.

Exercise 3.3-2:

*Answer **briefly** the following questions:*

1. *On which port listen server applications using SSL?*
2. *Which are the main components (subprotocols) of the SSL protocol?*

3.3.3 Transport Layer Security (TLS)

IETF TLS WG

TLS is an IETF standardisation initiative whose goal is to produce an Internet standard version of SSL. The IETF Transport Layer Security Work Group (TLS WG) was founded in April 1996 with the goal to write Internet standard track RFCs for a TLS protocol using the currently available specifications of SSL (2.0 and 3.0), PCT (1.0) and SSH (2.0) as basis. In December 1996, a first TLS 1.0 document was released as an Internet Draft (RFC 2246) and this specification of TLS 1.0 is conceptually similar to SSL 3.0.

Goals of TLS protocol

The goals of TLS Protocol, in order of their priority, are:

1. **Cryptographic security**: TLS should be used to establish a secure connection between two parties.
2. **Interoperability**: Independent programmers should be able to develop applications utilizing TLS that will then be able to successfully exchange cryptographic parameters without knowledge of one another's code.
3. **Extensibility**: TLS seeks to provide a framework into which new public key and bulk encryption methods can be incorporated as it is necessary. This will also accomplish two sub-goals: to prevent the need to create a new protocol (and risking the introduction of possible new weaknesses) and to avoid the need to implement an entire new security library.
4. **Relative efficiency**: Cryptographic operations tend to be highly CPU intensive, particularly public key operations. For this reason, the TLS protocol has incorporated an optional session caching scheme to reduce the number of connections that need to be established from scratch. Additionally, care has been taken to reduce network activity.

TLS record protocol

Similar to SSL, the TLS protocol is a layered protocol. On the lower layer, the **TLS record protocol** takes messages to be transmitted, fragments the data into manageable blocks, optionally compresses them, computes and appends a MAC to each block, encrypts the result and transmits it. On the receiver side, a received TLS ciphertext record is decrypted, verified, decompressed, and reassembled, before it is delivered to a higher level client. A TLS connection state is the operating environment of the TLS record protocol. It specifies compression, encryption, and MAC algorithms, and determines parameters for these algorithms (encryption and MAC keys, initialisation vectors). The TLS record format is the same as that of SSL record format (Fig. 3.3-8), and the fields in the header have the same meanings. The only differences are in the version values.

TLS Handshake Protocol

On the higher layer, several TLS subprotocols operate: TLS Handshake Protocol, TLS Change Cipher Spec Protocol, and TLS Alert Protocol. These protocols are similar to the corresponding SSL protocols. The **TLS Handshake Protocol** is used to negotiate a session state that consists of a session identifier, a peer certificate, a compression method, a cipher spec, a master key and a flag whether the session is resumable and can be used to initiate new connections. These items are used to create security parameters for use by the TLS record protocol when protecting application data. The corresponding message flow in the TLS Handshake Protocol is essentially the same as the message of the SSL handshake protocol discussed previously.

TLS Change Cipher Spec Protocol

TLS alert protocol

The **TLS Change Cipher Spec Protocol** consists of a single Change Cipher Spec message that is sent by the client and server during the handshake. The **TLS alert protocol** is used to send alert messages that convey the severity of a message and description of the alert. Like other messages, alert messages are compressed, authenticated, and encrypted as specified by the current connection state.

Differences between TLS and SSL

The differences between TLS 1.0 and SSL 3.0 are not huge,but they are significant enough that TLS 1.0 and SSL 3.0 do not easily interoperate. Nevetheless, TLS 1.0 does incorporate a mechanism by which a TLS implementation can back down to SSL 3.0.

The three major modifications suggested to SSL 3.0 to be incorporated into TLS 1.0 are:

- The HMAC construction developed in the IETF IPSec WG should be adopted and used in TLS 1.0.
- The Fortezza key exchange algorithm should be removed from TLS 1.0 since it refers to a proprietary and unpublishad technology.
- In comparison to SSL, TLS introduced more separation and a formal interface between the handshaking process and the record layer. The idea is to make these to related but separate standards.

There are two differences between SSL 3.0 and TLS MAC schemes: the actual algorithm and the scope of the MAC calculation. TLS makes use of the HMAC algorithm. Recall that HMAC of message `M`, based on hash algorithm `H` and using key `K` is defined as:

$$HMAC(K, M) = H(KXORopad, H(KXORipad, M)),$$

where `ipad` = 00110110 repeated 64 times (512 bit) and `opad` = 01011100 repeated 64 times (512 bit). SSL 3.0 uses the same algorithm, except that the padding bytes are concatenated with the secret key rather than being XORed with the secret key padded to the block length. The MAC calculation in TLS covers all the fields covered by the SSL 3.0 calculation, plus the version of the protocol being employed.

TLS supports all of the alert codes defined in SSL 3.0 with the exception of `no_certificate`. Moreover, a number of additional codes are defined in TLS. There are also several small differences between the cipher suites available under TLS and SSL 3.0. As mentioned, TLS supports all of the key exchange techniques of SSL 3.0 with the exception of Fortezza. TLS also includes all of the symmetric encryption algorithms found in SSL 3.0 with the exception of Fortezza. We do not adress here other slight differences in the certificate types, padding and cryptographic computations.

For historical reasons and in order to avoid a profligate consumption of reserved port numbers, application protocols which are secured by SSL 3.0 and SSL 2.0 all frequently share the same connection port: for example, the https protocol (HTTP secured by SSL or TLS) uses port 443 regardless of which security protocol it is using. Thus, some mechanism must be determined to distinguish and negotiate among the various protocols.

Used ports

Development and improvements

TLS 1.0 has been submitted to the IESG[22] for consideration as proposed standard. In addition, there are proposals to improve and further enhance the TLS protocol. One proposal, for example, is to use Kerberos as an additional authentication method. Another proposal is to fit password-based authentication into the TLS protocol (the reasoning is that passwords are still in widespread use today, as opposed to public key certificates). Current working items of IETF TLS WG also address elliptic curve cryptosystems cipher suites, the use of HTTP over TLS, and other refinements.

Security problems of UDP-based applications

Unfortunately, the protocols have some problems, which we address here. First, neither SSL nor TLS provide a viable solution for the security related problems of UDP-based applications. They only support security services for connection-oriented transport, such as those provided by TCP over IP. The SSL and TLS protocols both require a TCP connection before the handshake may be performed. Consequently, if no TCP connection is established at all (as in the case of UDP-based applications), neither the SSL nor the TLS protocol can be used. In the long term, this may pose some problems, since an increasingly large number of applications (realtime communications, multicast applications) make use of UDP as a transport layer protocol. Until the UDPsecurity is addressed, one can secure it at the Internet layer (with IPSec) or at the application layer using an application-specific security enhancement.

It is paramount to comprehend that TSL (or SSL) do not automatically offer security. For TSL (or SSL) to be able to provide a secure connection, more is needed than just flawless protocol. The server and client side systems have to be secure or the whole system is compromised. The same applies to everything else (keys, application programs). The greatest challenge lies not in the TSL (SSL) protocol itself, but rather in bug free implementation of it and in the development of secure, well-behaving applications on top of it.

Exercise 3.3-3:

Answer ***briefly*** *the following questions:*

1. *What is TLS?*
2. *On which port listen server applications using TLS?*
3. *Which are the main components (subprotocols) of the TLS protocol?*

22 Internet Engineering Steering Group

3.3.4 Summary

Implementation possibilities

Both SSL and TLS transport layer security protocols are well suited to provide communicaton security services for TCP-based applications. In fact, the user community of the SSL and TLS protocols is growing very rapidly. There are two implementation choices. For full generality, SSL or TLS could be provided as part of the underlying protocol suite and therefore be transparent to applications. Alternatively, SSL can be embedded in specific packages. For example, Netscape and Microsoft Explorer browsers come equipped with SSL, and most Web servers have implemented the protocol.

3.4 Security at the Application Layer

The application layer provides the services for an application to send and receive data over the network. It also provides services such as name resolution (refer to DNS). Applications such as World Wide Web (WWW) browers or email clients use the services provided by the application layer to communicate with its peers, WWW servers and e-mail servers respectively. The application layer also defines the interface to the transport layer.

Advantages and disatvantages

Application-level security has to be implemented in end hosts. Providing security at the application layer is often the most intrusive option. It is also the most flexible, because the scope and the strength of the protection can be tailored to meet the specific needs of the application and the user has complete access to the data he wants to protect and to credentials such as private keys. An application can be extended without having to depend on the operating system to provide these services (normally, applications have no control over what is implemented in the operating system). The downside of application layer security is that the security mechanisms have to be designed independently for each application. This implies existing applications have to be enhanced to provide security. As each application has to define its own security mechanisms, there is a greater probability of making mistakes and hence opening up security holes for attacks.

Approaches

In general, there are two **approaches** to provide security services at the application layer: either the services are integrated into each application protocol individually (security-enhanced application protocols), or a generic security system is built that can be used to incorporate security services into arbitrary applications (authentication and key distribution systems).

Several application protocols have been enhanced to provide integrated security services. There are security enhencements for Telnet (STEL), for electronic mail (PEM, PGP, S/MIME), for WWW transactions (Secure-HTTP, SET) and some other applications. Also, considerable amount of work has been spent in the past few years to develope authentication and key distribution systems that can be used by arbitrary applications to incorporate security services. Examples are Kerberos, NetSP (Network Security Program), and others. Since later Chapters of this course are devoted to application layer security, we will not further discuss this mechanisms here.

3.5 Summary

As it has been presented above, in today’s Internet, there are a lot of protocols designed to secure traffic at various levels in the network. In general, it is difficult to say what layer is best suited to address the security requirements of specific applications and application protocols. The answer to this question heavily depends on the application. The advantage of placing security lower in the protocol stack is that at this layers security can be implemented transparently to users and application programs. Puting the security on the higher layers has its own advantage, because only protocols that work on the application layer can meet application-specific security requirements.

Irrespective of where in the stack security is implemented, the basic services of confidentiality, integrity, authentication, authorisation, key management and non-repudiation have to be provided. Depending on where in the stack the security is implemented, it is possible to provide some or all of these services. In some cases, it does make sense to provide some capabilities at one layer and other capabilities at a different layer.

Exercise 3.5-1:

*Answer **briefly** the following questions:*

1. *Does the employing of some transport layer security protocol (SSH, SSL, or TLS) protect against traffic analysis attacks?*
2. *What is the main disadvantage of the SSL and TLS protocols?*
3. *What are the advantages and disadvantages of providing security mechanisms on the application layer?*

4 World Wide Web Security

4.1 Introduction

The World Wide Web was originally developed as a publishing medium for public documents and therefore provided few controls for restricting access to information. As a wider range of documents and services appeared on the web, improved security facilities to satisfy the new requirements were needed.

There are basically three overlapping types of security risks regarding the World Wide Web ([Ste98]):

1. Bugs or misconfiguration problems in the Web server that allow unauthorized remote users to:
   - Steal confidential documents.
   - Upload files on the server.
   - Execute commands on the server host machine, allowing them to modify the system.
   - Gain information about the Web server's host machine that will allow them to break into the system.
   - Launch denial-of-service attacks, rendering the machine temporarily unusable.
2. Browser-side risks, including:
   - Active content that crashes the browser, damages the user's system, breaches the user's privacy, or merely creates an annoyance.
   - The misuse of personal information knowingly or unknowingly provided by the end-user.
3. Interception of network data sent from browser to server or vice versa via network eavesdropping.

From the point of view of the network administrator, a Web server represents yet another potential hole in a local network's security. The general goal of network security is to keep strangers out, as a Web site provides the world with controlled access to a network. A poorly configured Web server can punch a hole in the most carefully designed firewall system, a poorly configured firewall can make a Web site impossible to use. Things get particularly complicated in an intranet environment, where the Web server must typically be configured to recognize and authenticate various groups of users, each with distinct access privileges.

To the end-user, Web surfing feels both safe and anonymous. It's not. Active content, such as ActiveX controls and Java applets, introduces the possibility that Web browsing will introduce viruses or other malicious software into the user's system.

Active content also has implications for the network administrator, insofar as Web browsers provide a pathway for malicious software to bypass the firewall system and enter the local area network. Even without active content, the very act of browsing leaves an electronic record of the user's surfing history, from which unscrupulous individuals can reconstruct a very accurate profile of the user's tastes and habits.

Finally, both end-users and Web administrators need to worry about the confidentiality of the data transmitted across the Web. The TCP/IP protocol was not designed with security in mind (for details see Chapter 3); hence it is vulnerable to network eavesdropping. When confidential documents are transmitted from the Web server to the browser, or when the end-user sends private information back to the server inside a fill-out form, someone may be listening in.

It is important to realize that ”secure” browsers and servers are only designed to protect confidential information against network eavesdropping. Without system security on both browser and server sides, confidential documents are vulnerable to interception.

Aspects regarding server-side security are the subject of Section 4.3, client-side security is covered in Section 4.4. We now consider authentication on the the Web

4.2 Authentication and the Web

In short, the process of verifying the identity of a requesting user is called user authentication, whereas the process of granting the privileges to access particular resources is called user authorization. In the simplest case the user is given a username and a password. The username identifies the person who wishes to access the HTTP server, and the password authenticates the person. To increase security, more sophisticated user authentication schemes may be used.

When using an FTP or telnet service, the user is authenticated during the login process, sends one or more commands to the service, and then logs out. The initial authentication remains in effect for all of the operations performed until the user logs out; the many interactions between user and server during this period are therefore regarded as a single session. The HTTP protocol has no concept of a session. Each connection which a user makes to a server carries only a single request and response and is independent of all other connections, even those between the same two parties. The protocol was designed in this way so that servers did not have to retain any information from one connection to the other; in particular this means that authentication information, if required, will not be retained by the server and must be included with every request. So long as the same authentication can be used for all documents on that server (or a known subset of those documents) this is not a problem as the necessary credentials can be stored by the browser program and reissued for each request. Any system which reuses the same credentials is vulnerable to replay attacks but the alternative of using a one-time password token requires that the user enter a new password for every request. Since each displayed page may involve several requests (every graphic on the page requires a separate request) this would soon become intolerable.

In Section 4.3.1 we address two user authentication schemes that have been proposed for HTTP: basic authentication and digest authentication. They both implement a password-based protection scheme for HTTP.

4.3 Server-side security

Web servers are usually run to make resources publicly available and accessible to users. Therefore, the Web servers are typically configured to accept requests from anonymous users, and there is no need for user authentication and authorisation. If organisations run Web servers whose resources may be restricted to employees of the organisation or to customers who have paid a subscription fee, proper access control is needed (Section 4.3.1).

CGI scripts also introduce serious security concerns, since one way to break into a Web server is to exploit a weakness in the CGI programs that are accessible through the Web server itself. CGI scripts are subject of Section 4.3.2.

Finally, the Internet carries an increasing amount of private traffic and all requests for documents are logged by the Web server. Server logs and privacy are subject of Section 4.3.3.

4.3.1 Authentication and access restriction

In general, there are several techniques that may be employed to control access to resources on Web servers ([Ste98]):

1. Restriction by IP address, subnet, or domain:
   Individual documents or whole directories are protected in such a way that only browsers connecting from certain IP (Internet) addresses, IP subnets, or domains can access them.

2. Restriction by user name and password:
   Documents or directories are protected so that the remote user has to provide a name and a password in order to get access.

IP address or domain name

Most Web servers allow their administrators to restrict access to a particular group of computers based on the computers' IP address or DNS hostname.

Restriction by IP address is secure against curiousness but not against a determined hacker. There are several ways around IP address restrictions. With the proper equipment and software, a hacker can ”spoof” the IP address, making it seem as if he is connecting from a location different from his real one. Moreover, there is no guarantee that the person contacting your server from an authorized host is in fact the person you think he is. The remote host may have been broken into and is being used as a front. To be safe, IP address restriction must be combined with something that checks the identity of the user, such as a check for user name and password.

IP address restriction can be made much safer by running your server behind a firewall machine (see Chapter 6) that is capable of detecting and rejecting attempts at spoofing IP addresses. Such detection works best for intercepting packets from the outside world that claim to be from trusted machines on your internal network.

One thing to be aware of is that if a browser is set to use a proxy server (see Section 6.3) to fetch documents, then your server will only know about the IP address of the proxy, not the real user's. This means that if the proxy is in a trusted domain, anyone can use that proxy to access your site. Unless you know that you can trust a particular proxy to impose its own restriction, do not add the IP address of a proxy (or a domain containing a proxy server) to the list of authorized addresses.

Restriction by host or domain name has the same risks as restriction by IP address, but also suffers from the risk of ”DNS spoofing”, an attack in which your server is temporarily fooled into thinking that a trusted host name belongs to an alien IP address. To reduce that risk, some servers can be configured to do an extra DNS lookup for each client. After translating the IP address of the incoming request to a host name, the server uses the DNS to translate from the host name back to the IP address. If the two addresses don't match, the access is denied.

The details of how to restrict access to documents by the IP address or domain name of the remote browser are different for each server. For Apache HTTP servers, you will need to add a directory control section to the file access.conf (or to the file .htaccess) that looks something like this:

```
<Directory /full/path/to/directory>
<Limit GET POST>
order mutual-failure
deny from all
allow from 134.147. .fernuni-hagen.de
allow from 130.12.14.15 host35.siemens.de
</Limit>
</Directory>
```

This will deny access to everybody but the indicated hosts (`130.12.14.15` and `host35.siemens.de`), subnets (`134.147.`) and domains (`.fernuni-`

`hagen.de`). Although you can use either numeric IP addresses or host names, it is safer to use the numeric form because this form of identification is less easily subverted.

HTTP basic authentication

The method used in HTTP for user authentication is quite simple. Since HTTP is a stateless protocol - that is, the server does not remember any information about a request once it has finished - the browser needs to resend the username and password on each request.

On the first access to an authenticated resource, the server will return a 401 status (”Unauthorized”) and include a WWW-Authenticate response header. This will contain the authentication scheme to use and the realm name. The browser would then ask the user to enter a username and a password. It then requests the same resource again, this time including an authorization header which contains the scheme name (”Basic”) and the username and password entered. The server checks the username and password, and if they are valid, returns the page. If the password is not valid for that user, or the user is not allowed access because he is not listed on a require user line or in a suitable group, the server returns a 401 status as before. The browser can then ask the user to retry hisusername and password.

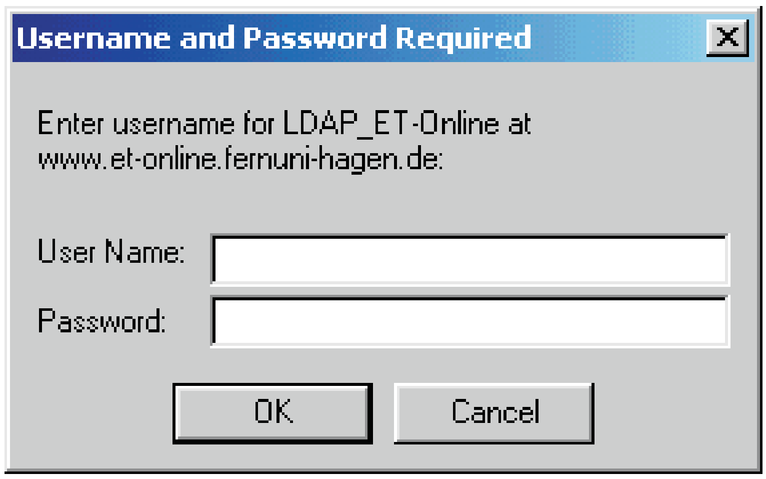

Username and Password Required
Enter username for LDAP_ET-Online at www.et-online.fernuni-hagen.de:
User Name:
Password:
OK
Cancel

Fig. 4.3-1: Netscape Communicator 4.76 Username and Password Required prompt.

Assuming the username and password was valid, the user might next request another resource which is protected. In this case, the server would respond with a 401 status, and the browser could send the request again with the user and password details. However this would be slow, so instead the browser sends the Authorization header on subsequent requests. Note that the browser must ensure that it only sends the username and password to further requests on the same server (it would be insecure to send these details if the user moved onto a different server).

The browser needs to remember the username and password entered, so it can send these with future requests from the same server. Note that this can cause problems when testing authentication, since the browser remembers the first username and password that works. It might be difficult to force the browser to ask for a new username and password.

HTTP digest authentication

The HTTP basic authentication is not considered to be a secure method of user authentication, as the user name and password are passed over the network in an unencrypted form.

Version 1.1 of HTTP (published in July 1996) introduced an improved method called digest authentication. This uses the same exchange of packets as basic authentication, but the ”Unauthenticated” reply now includes a value, known as the nonce, which acts as a challenge. Instead of replying with the username and password, the client calculates a message digest (using the MD5 algorithm) from the username, password and nonce and returns this along with the username as authentication information. The server then repeats the MD5 calculation, using the user’s correct password, and returns the document if the two digests match. To do this the server must hold each user’s password in a form suitable for calculating the MD5 digest. It is imperative that these passwords be stored as securely as possible, since anyone possessing them would immediately be able to masquerade as any valid user of the server. The need for security is even greater than for a Unix password file, whose contents must be de-crypted by brute force methods before they can be misused.

The HTTP server is stateless so it cannot ”remember” the nonce value between each challenge and its response; the nonce must therefore be derived from some combination of information from the request packet along with values held centrally on the server. This still leaves the process vulnerable to replay attacks as the server does not ensure that nonces are unique to a single request. However by careful choice of the nonce this risk and the resulting damage can be substantially reduced. A good nonce calculation will usually include the URL of the document requested so that a successful replay attack can only retrieve a single document, rather than the whole realm as with basic authentication. The IP address of the client making the request should also be part of the calculation: a replay attack from a different client machine will then fail unless the intruder has changed his IP address and managed to persuade the routing systems to return responses to the new location. The server may also maintain a single ’magic number’ which changes periodically. Including this in the nonce will limit the time period during which a replay attack can be successful. However the rate of change must be sufficient to allow a legitimate user to enter a username and password as otherwise the nonce will have expired by the time the authenticated request is sent. To assist the user, it is suggested that servers should recognise digests which are correct except for using an ’old’ nonce and request that the browser program re-calculate the digest with the latest value. This second calculation can be done without consulting the user (since the browser will have cached the username and password) and as such should have a shorter round-trip time.

Only the server needs to know how the nonce is calculated, as such different servers can choose methods appropriate to the sensitivity of the information they hold. Each improvement in security requires more processing in the calculation of the nonce and the message digest; for a busy server this may represent a considerable load. A nonce with a short lifetime is also likely to require many re-authentications, increasing the network traffic. The only way to prevent replay attacks is to keep a record of all digest values received by the server, and ensure that each one is only used once. However this involves the server holding a great deal of information which must be checked at every request. There is still no protection against other types of attacks, for example through a compromised or hostile proxy between the client and the server. These can only be prevented by cryptographic encryption of the request and response so that even if transactions fall into wrong hands, they are of no use. The standard concludes: ”Digest Authentication does not provide a strong authentication mechanism. That is not its intent. It is intended solely to replace a much weaker and even more dangerous authentication mechanism: Basic Authentication. An important design constraint is that the new authentication scheme be free of patent and other export restrictions. Most needs for secure HTTP transactions cannot be met by Digest Authentication. For those needs SSL or SHTTP are more appropriate protocols.”

SSL can be used both for server-side and for client-side authentication. For detailed information on SSL see Section 3.3.2.

Certificate-based authentication

As presented in Section 3.3.2 and Section 3.3.3, SSL and TLS can be used to secure communications between browsers and HTTP servers. Using HTTP on top of SSL or TLS makes user authentication and authorisation simple and straightforward. In short, Web servers authenticate themselves using site certificates, whereas individual users authenticate themselves using personal certificates.

In addition to superior security, there is another advantage related to the use of site and personal certificates. A certificate generally binds a public key to an identity. This identity can, in turn, be used to control access to system resources.

.htaccess password protection

If the users who are allowed to access restricted resources are widely dispersed, or the server administrator needs to be able to control access on an individual basis, it is possible to require a username and a password before being granted access to a document.

Setting up user authentication takes two steps (in the following described for Apache HTTP servers):

1. Create a file containing the usernames and passwords.

2. Tell the server what resources are to be protected and which users are allowed (after entering a valid password) to access them.

Creating a user database

A list of users and passwords needs to be created in a file. For security reasons, this file should not be under the document root. The examples here will assume you want to use a file called users in your server root at `/usr/local/etc/httpd`.

The file will consist of a list of usernames and a password for each. The format is similar to the standard Unix password file, with the username and password being separated by a colon. However you cannot just type in the usernames and passwords because the passwords are stored in an encrypted format. The program `htpasswd` is used to create a user file and to add or modify users.

`htpasswd` is a C program that is supplied in the support directory of the Apache distribution. If it is not already compiled, you will have to compile it first. Run `make htpasswd` in the support directory to compile it (you might need to modify the `makefile` first, since any configuration you did when compiling the server itself is not available to this `makefile`). After compilation, you can either leave the `htpasswd` binary where it is, or move it to a directory on your path (e.g. `/usr/local/bin`). In the former case, you will need to remember to give the full pathname to run it. The examples here will assume that it is installed somewhere on your path.

Using htpasswd

To create a new user file and add the username bob with the password `crypto01` to the file `/usr/local/etc/httpd/users`:

```
htpasswd -c /usr/local/etc/httpd/users bob
```

The `-c` argument tells `htpasswd` to create a new users file. When you run this command, you will be prompted to enter a password for bob, and confirm it by entering it again. Other users can be added to the existing file in the same way, except that the -c argument is not needed. The same command can also be used to modify the password of an existing user.

After adding a few users, the `/usr/local/etc/httpd/users` file might look like this:

```
bob_wRu808bHQAi48
jane:IabCqFQS59e8m
alice:FADHn3w774Ssu
```

The first field is the username, and the second field is the encrypted password.

Configuring the server

To get the server to use the usernames and passwords in this file, you need to configure a realm. This is a section of your site that is to be restricted to some or all of the users listed in this file. This is typically done on a per-directory basis, with a directory (and all its subdirectories) being protected (Apache 1.2 and later also let you protect individual files). The directives to create the protected area can be placed in a `.htaccess` file in the relevant directory, or in a `<Directory>` section in the `access.conf` file.

To allow a directory to be restricted within a `.htaccess` file, you first need to ensure that the `access.conf` file allows user authentication to be set up in a `.htaccess` file. This is controlled by the `AuthConfig` override. The `access.conf` file should include `AllowOverride AuthConfig` to allow the authentication directives to be used in a `.htaccess` file.

To restrict a directory to any user listed in the users file just created, you should create a `.htaccess` file containing:

```
AuthName "restricted stuff"
AuthType Basic
AuthUserFile /usr/local/etc/httpd/users

require valid-user
```

The first directive, `AuthName`, specifies a realm name for this protection. Once a user has entered a valid username and password, any other resources within the same realm name can be accessed with the same username and password. This can be used to create two areas which share the same username and password.

The `AuthType` directive tells the server what protocol is to be used for authentication. At the moment, HTTP basic is the only method available. However a new method, HTTP digest, is about to be standardised, and once browsers start to implement it, HTTP digest authentication will provide more security than the HTTP basic authentication.

`AuthUserFile` tells the server the location of the user file created by `htpasswd`. A similar directive, `AuthGroupFile`, can be used to tell the server the location of a group file (see below).

These four directives tell the server where to find the usernames and passwords and what authentication protocol to use. The server now knows that this resource is restricted to valid users. The final stage is to tell the server which usernames from the file are valid for particular access methods. This is done with the require directive. In this example, the argument valid-user tells the server that any username in the users file can be used. But it could be configured to allow only certain users in:

```
require user bob jane
```

would only allow users bob and jane access (after they entered a correct password). If user alice (or any other user) tried to access this directory - even with the correct password - they would be denied. This is useful to restrict different areas of your server to different people with the same users file. If a user is allowed to access different areas, he only has to remember a single password. Note that if the realm name differs in the different areas, the user will have to re-enter his password.

Using groups

If you want to allow only selected users from the users file in to a particular area, you can list all the allowed usernames on the `require` line. However this means you are building username information into your `.htaccess` files, and this might not be convenient if there are a lot of users. Fortunately there is a way round this, using a `group` file. This operates in a similar way to standard Unix groups: any particular user can be a member of any number of groups. You can then use the `require` line to restrict users to one or more particular groups. For example, you could create a group called `staff` containing users who are allowed to access internal pages. To restrict access to just users in the `staff` group, you would use

```
require group staff
```

Multiple groups can be listed, and `require user` can also be given, in which case any user in any of the listed groups, or any user listed explicitly, can access the resource. For example

```
require group staff admin
require user adminuser
```

which would allow any user in group `staff` or group `admin`, or the user `adminuser`, to access this resource after entering a valid password.

A group file consists of lines giving a group name followed by a spaceseparated list of users in that group. For example:

```
staff:bob jane
admin:alice adminuser
```

The `AuthGroupFile` directive is used to tell the server the location of the group file. Note that the maximum line length within the group file in about 8000 characters. If you have more users in a group than will fit within that line length, you can have more than one line with the same group name within the file.

Other ways of storing user details

While Apache by default can only access user details in plain text files, various add-on modules are available to allow user details to be stored in databases. Besides DBM format, user and group lists can be stored in DB format files. Or full databases can be used, such as `mSQL`, `Postgres95` or any DBI-compatible database.

It is also possible to have an arbitrary external program check whether the given username and password is valid (this could be used to write an interface to check against any other database or authentication service). Modules are also available to check against the system password file, or to use a Kerberos system (see Chapter 6)
.

Limiting methods differently

In the example `.htaccess` file above, the require directory is not given inside a `<Limit>` section. This is valid in Apache, and means that it applies to all request methods. In other servers and most example `.htaccess` files, the `require` directive is given inside a `<Limit>` section, such as this:

```
<Limit GET POST PUT>
require valid-user
</Limit>
```

In Apache it is better to omit the `<Limit>` and `</Limit>` lines, to ensure that the protection applies to all methods. However, this format can be used to limit particular methods. For example, to limit just the `POST` method, use

```
AuthName "restrict posting"
AuthType Basic
AuthUserFile /usr/local/etc/httpd/users

<Limit POST>
require group staff
</Limit>
```

Now only members of the group staff will be allowed to `POST`. Other users (unauthenticated) can use other methods, such as `GET`. This could be used to allow a `CGI program` to be accessed by anyone, but only authorised uses can `POST` information to it.

4.3.2 CGI scripts

CGI is short for Common Gateway Interface. In common usage CGI means both a kind of programming interface and the idea of enabling a given URL to return something dynamic rather than something static. CGI is a sort of meta-language, or middleware, as the term is generally used. Its syntax is loosely de facto standard handed down from the NCSA server, but CGI has had a number of proprietary ”enhancements” here and there. Nonetheless, the core idea is sound and proven - a programming language for scripts that are likely to run unmodified on a good variety of similar platformas, all UNIX platforms, for example.

But CGI is not a full programming language itself in the way that C is a programming language; it is just a fairly platform-independent way to reach a runnable program from a URL. ”Runnable programs reachable via CGI” can be in just about any language that the local platform can understand, but some are standard across all platforms. The only requirements on a CGI language are that it be able to get at the same environment variables as the `httpd` and that it be able to read from `stdin` and `stdout` (e.g. `Perl`, `Tcl`, and `Java`).

Because CGI talks to programs, naturally it has arguments to send and return-values to receive. Both these are carried either in the URL or in HTML forms.

This conversation by way of URL can take one of two forms; it is either a keyword list or a named parameter list. For a keyword list, the parameters are text set off from each other by the plus sign (+) and from the body of the URL by the question mark (?). For a named parameter list, the parameters are a concatenated set of attribute-value pairs set off from each other by the ampersand (&) and from the URL body by the question mark (?). Examples might look like this:

```
 ... naturally it has a set of
<A HREF="/rub2/chapter4/attribute+value">
attribute-value pairs
</A> set off from each other ...
```

and this:

```
 Examples might look like
<A HREF="/chapter4/index/figures?entry=examples&chapter=6>
this
</A>
```

You either configure your server so that a particular directory contains all your CGI programs and only your CGI programs or you globally declare that a certain filename extension always identifies a runnable script, or both. The typical installed Apache server will be looking for the subdirectory `./cgi-bin` in the server root or for filenames ending in `.cgi`.

CGI scripts are a major source of security holes. Although the CGI protocol is not inherently insecure, CGI scripts must be written with just as much care as the server itself. Unfortunately some scripts fall short of this standard.

CGI scripts can present security holes in two ways:

1. They may intentionally or unintentionally leak information about the host system that will help hackers break in.

2. Scripts that process remote user input, such as the contents of a form or a ”searchable index” command, may be vulnerable to attacks in which the remote user tricks them into executing commands.

Although there is nothing intrinsically dangerous about scattering CGI scripts around the document tree, it is better to store them in the `cgi-bin` directory. Because CGI scripts are such potentially large security holes, it is much easier to keep track of which scripts are installed on your system if they are kept in a central location rather than being scattered around among multiple directories. This is particularly true in an environment with multiple Web authors. It is just too easy for an author to inadverently create a buggy CGI script and install it somewhere in the document tree. By restricting CGI scripts to the `cgi-bin` directory and by setting up permissions, so that only the Web administrator can install these scripts, you avoid this chaotic situation.

There is also a risk of a hacker managing to create a `.cgi` file somewhere in your document tree and then executing it remotely by requesting its URL. A `cgi-bin` directory with tightly-controlled access lessens the possibility of this happening.

4.3.3 Server logs and privacy

Most servers log every access. The log usually includes the IP address and/or host name, the time of the download, the user’s name (if known by user authentication or obtained by the idented protocol), the URL requested (including the values of any variables from a form submitted using the GET method), the status of the request, and the size of the data transmitted. Some browsers also provide the client the reader is using, the URL that the client came from, and the user’s e-mail address. Servers can log this information as well, or make it available to CGI scripts. Most WWW clients are probably run from single-user machines, thus a download can be attributed to an individual. Revealing any of these datums could be potentially damaging to a reader.

Another way Web usage can be revealed locally is via browser history, hotlists, and cache. If someone has access to the reader’s machine, he can check the contents of these databases. An obvious example is shared machines in an open lab or public library.

Proxy servers used for access to Web services outside an organization’s firewall are in a particularly sensitive position. A proxy server will log every access to the outside Web made by every member of the organization and track both the IP number of the host making the request and the requested URL. A carelessly managed proxy server can therefore represent a significant invasion of privacy.

All requests for documents are logged by the Web server. Although your name is not usually logged, your IP address and computer's host name usually is. In addition, some servers also log the URL you were viewing (such as your home page) at the time you requested the new URL. If the site is well administered, the record of your accesses will be used for statistics generation and debugging only. However, some sites may leave the logs open for casual viewing by local users at the site or even use them to create mailing lists.

The contents of queries in forms submitted using the GET request appear in the server log files because the query is submitted as part of the URL. However, when a query is submitted as a POST request (which is often the case when submitting a fill-out form), the data you submit does not get logged. If you are concerned about the contents of a keyword search appearing in a public log somewhere, check whether the search script uses the GET or the POST method. The easiest technique is to try an innocuous query first. If the contents of the query appear in the URL of the retrieved document, then they probably appear in the remote server's logs too.

Server/browser combinations that use data encryption, such as Netsite/ Netscape, encrypt the URL request. Furthermore the encrypted request, because it is submitted as a POST request, does not appear in the server logs.

4.4 Client-side security

One of the most useful and powerful feature of browsers is their ability to download executable content from remote sites. Java, JavaScript, and ActiveX are examples of executable content that can be downloaded to a machine running a browser. Java, in particular, can be used to run platform-independent applications that can be prototyped quickly. One of the features of the Java interpreter in Netscape's Navigator and Microsoft's Internet Explorer is that applets cannot leave the controlled environment. This is equal to claiming that Java is secure which is far from true (see Section 4.4.3).

ActiveX is different from the Java model: It assumes that content is digitally signed by trusted distributors. The client blindly trusts the signed code and no further security measures are taken. One problem with this model is that there is no global public key infrastructure to scale the solution to the Web.

In this chapter, we describe the security of SSL (Section 4.4.1), the security problems of JavaScript (Section 4.4.2), and ActiveX (Section 4.4.4). Next, we explore Java and the Java environment (Section 4.4.3). Finally, we discuss cookies (Section 4.4.5).

4.4.1 SSL

SSL uses public-key encryption to exchange a session key between the client and the server; this session key is used to encrypt the http transactions (both request and response). Each transaction uses a different session key so that if someone manages to decrypt a transaction, it does not mean that he has found the server's secret key; if he wants to decrypt another transaction, he will need to spend as much time and effort on the second transaction as he did on the first.

Netscape servers and browsers do encryption using either a 40-bit secret key or a 128-bit secret key. Many people feel that using a 40-bit key is insecure because it is vulnerable to a ”brute force” attack (trying each of the 2^{40} possible keys until you find the one that decrypts the message). This was in fact demonstrated in 1995 when a French researcher used a network of workstations to crack a 40-bit encrypted message in a little over a week. It is thought that with specialised hardware, 40-bit messages can be cracked in minutes to hours. Using a 128-bit key eliminates this problem because there are 2^{128} instead of 2^{40} possible keys. To crack a message encrypted with such a key by brute force would take significantly longer than the age of the universe using conventional technology.

In Netscape versions 3.X and earlier you can tell what kind of encryption is in use for a particular document by looking at the ”document information” screen accessible from the file menu. The little key in the lower left-hand corner of the Netscape window also indicates this information. A solid key with three teeth means 128-bit encryption, a solid key with two teeth means 40-bit encryption, and a broken key means no encryption. Even if your browser supports 128-bit encryption, it may use 40-bit encryption when talking to older servers.

In Netscape versions 4.X and higher, click on the ”Security” button to determine whether the current page is encrypted, and, if so, what level of encryption is in use. In Microsoft Internet Explorer, a solid padlock will appear on the bottom right of the screen when encryption is in use. To determine whether 40-bit or 128-bit encryption is in effect, open the document information page using `File->Properties`. This will indicate whether ”weak” or ”strong” encryption is in use.

4.4.2 JavaScript

JavaScript is a scripting language developed by Netscape[23]. Although it shares many of the features and structures of the full Java language, it was developed independently and is endorsed by a number of software companies. JavaScript is an open language that anyone can use without purchasing a license. It is supported

23 There are several books describing the JavaScript language in detail (e.g. [Fla96]).

by recent browsers from Netscape and Microsoft, though Internet Explorer supports only a subset, which Microsoft calls Jscript[24].

JavaScript enables Web authors to allow code to be contained within HTML documents themselves. This code could dynamically change the HTML that the browser interprets based on many conditions. For example, a JavaScript code segment could change the users display in his browser based on the values of the cookies set on the machine. JavaScript is especially well suited for popping up dilogue boxes and receiving input from the users.

One of JavaScript's most useful features is its ability to define userspecified event handlers. These can cause certain pieces of code to execute in response to events such as mouse positioning or keystroke entries. In addition, as JavaScript is a complete programming language, mathematical calculations and all sorts or algorithms can be implemented.

Security model

Java scripts, because they execute on the browser's side of the connection instead of on the server's, move the security risk from the server to the client. Many of JavaScript's security problems cannot be exploited directly, since they require user interaction. However, fooling users into providing the assistance necessary to mount the attacks is easy. For example, many attacks require that a user clicks on a button to activate the malicious code. A simple trick ist to use JavaScript to display a pop-up dialogue box and get the user to click on a button. One way to do this is to display a dialogue box with one button labeled OK. The unknowing user clicks the button, and the attack is activated.

JavaScript has a troubling history of security holes. Here are a few examples:

- Ability to read arbitrary files on user's machine (November 1998):
  A bug in the JavaScript implementation in Netscape Communicator 4.5 and 4.04-4.05 allows a Web page to read arbitrary files from the user's machine and transmit them across the Internet. Any file that can be read with the user's permission is vulnerable, including the system password file. The bug affects both Windows and Unix versions of Communicator. Any HTML page can carry this exploit, including ones that are transmitted as an e-mail enclosure. Internet Explorer has not been reported to be vulnerable.

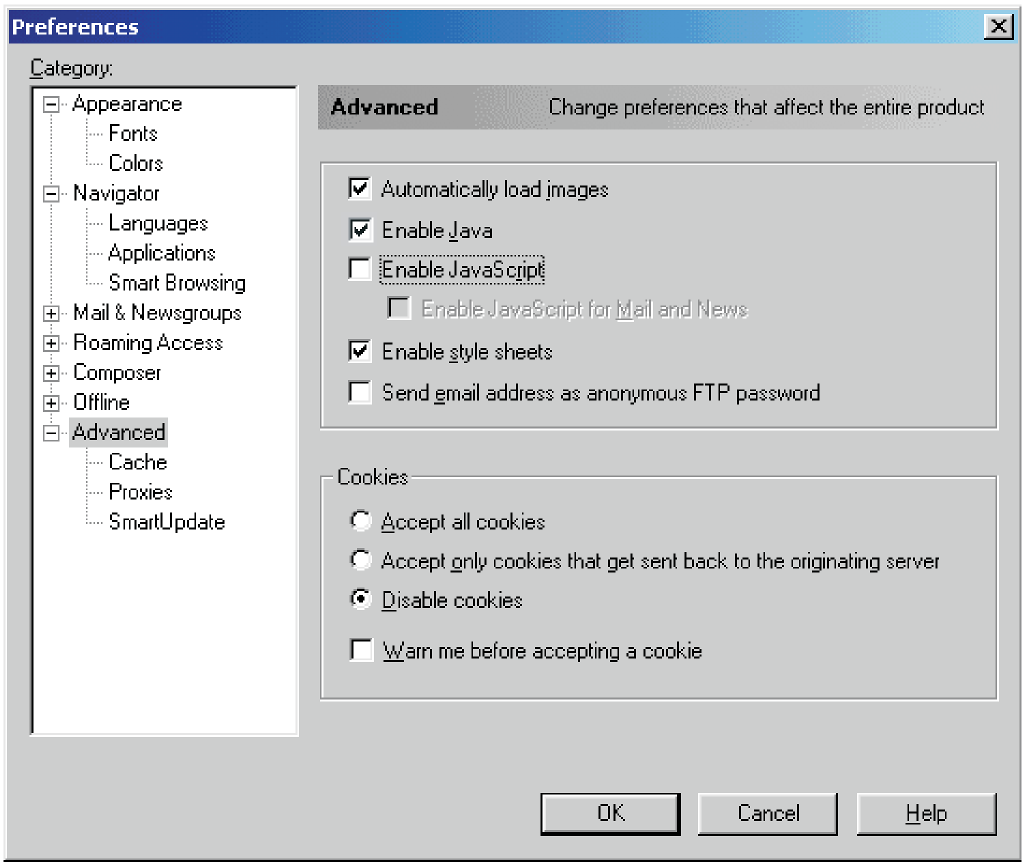
- The "Cuartango" and "Son of Cuartango" holes (November 1998):
  Microsoft Internet Explorer is also vulnerable to file theft via Java- Script. Internet Explorer versions 4.0-4.01 and prerelease versions of IE 5.0 allow JavaScript

24 JScript is Microsoft's extended implementation of ECMAScript (ECMA262), an international standard based on the Netscape's JavaScript and Microsoft's JScript languages. JScript is implemented as a Windows Script engine. This means that it can be "plugged in" to any application that supports Windows Script, such as Internet Explorer, Active Server Pages, and Windows Script Host. It also means that any application supporting Windows Script can use multiple languages - JScript, VBScript, Perl, and others.

programmes to cut and paste text into file upload fields, thereby allowing a boobytrapped Web page or e-mail message to steal any file on the user's disk.

- Ability to intercept the user's e-mail address and other preferences (February 1998):
  Versions of Netscape Navigator 4.0 through 4.04 contain a security hole involving access of JavaScript programmes to the browsers preferences settings. The settings, which are stored in a file named `preferences.js` (or `prefs.js`) in the Netscape directory, include a variety of private information such as your email address, the names of mailbox files, and the identity of your e-mail and news servers. In addition, in many cases the preferences file also stores your e-mail (POP) and FTP (publish) passwords. To see what else is in this file, open it in a text editor and take a look. The implications of this hole is that a JavaScript enabled page can open the preferences file and upload all information contained within it to a remote server. This can be exploited to capture visitors' e-mail addresses and to gather information about the user's network configuration. The worst risk is that the user's e-mail password will be disclosed. Since the e-mail password is, in many cases, the same as the user's LAN login password, this exposes organizations to a potential route of attack.

People who worry about the disclosure of personal information are encouraged to turn off JavaScript completely. In Netscape Navigator 4.0 or higher, you can do this by deactivating a checkbox located in **Edit->Preferences->Advanced**. In Microsoft Internet Explorer versions 3.X, you can do this by unchecking a misleadingly-named checkbox labeled `Run ActiveX scripts` in **View->Options->Security**. In Microsoft Internet Explorer 4.0, the new `Security Zones` feature, which was designed to make the Internet safer, actually makes it difficult to turn off JavaScript, since it is active even when `High Security` is selected. To do this, go to **View->Internet Options->Internet Security**, and select the `Internet Zone`. Now select the radio button labeled `Custom` and press the adjacent `Settings...` button. Scroll down to the bottom of the option list, and disable the option labeled `Active Scripting`.


Preferences

Category:

- Appearance
  - Fonts
  - Colors
- Navigator
  - Languages
  - Applications
  - Smart Browsing
- Mail & Newsgroups
- Roaming Access
- Composer
- Offline
- Advanced
  - Cache
  - Proxies
  - SmartUpdate

Advanced Change preferences that affect the entire product

- ✓ Automatically load images
- ✓ Enable Java
- ☐ Enable JavaScript
- ☐ Enable JavaScript for Mail and News
- ✓ Enable style sheets
- ☐ Send email address as anonymous FTP password

Cookies

- ○ Accept all cookies
- ○ Accept only cookies that get sent back to the originating server
- ● Disable cookies
- ☐ Warn me before accepting a cookie

Fig. 4.4-1: How to disable JavaScript in Netscape Navigator 4.0 or higher.

4.4.3 Java

Java is a high-level programming language developed by Sun Microsystems[25]. Java was originally called OAK, and was designed for handheld devices and set-top boxes. Oak was unsuccessful so in 1995 Sun changed the name to Java and modified the language to take advantage of the growing World Wide Web.

Java is an object-oriented language similar to C++, but simplified to eliminate language features that cause common programming errors. Java source code files (files with a `.java` extension) are compiled into a format called bytecode (files with a `.class` extension), which can then be executed by a Java interpreter. Compiled Java code can run on most computers because Java interpreters and runtime environments, known as Java Virtual Machines (VMs), exist for most operating systems, including UNIX, the Macintosh OS, and Windows. Bytecode can also be converted directly into machine language instructions by a just-in-time compiler (JIT).

Small Java applications are called Java applets. Applets run from inside a web browser and HTML pages therefore contain an `<APPLET>` tag, which tells the browser where to find the Java `.class` files. For example,[26]

25 There are several books describing the Java language in detail (e.g. [Fla99]).

26 Elliptic Curves

```
<APPLET CODE=EllipticCurveCryptosystem.EllipticCurves.class
WIDTH=570 HEIGHT=400>
</APPLET>
```

Because applets are small in files size, cross-platform compatible, and highly secure (can't be used to access users' hard drives), they are ideal for small Internet applications accessible from a browser.

Security model

Since Java bytecodes can be generated without a compiler, restrictions on the Java compiler cannot be used for security. When we talk about Java security, we refer to access control on Java bytecodes that are received from an unknown source and executed. We would like these bytecode streams - called applets - to execute in a controlled environment from which they cannot escape. Such an environment is often termed as `sandbox`.

The idea behind the sandbox or virtual machine is that when running as applets, Java scripts are restricted with respect to what they are allowed to do by a `security manager` object. Things, that applets are prevented from doing:

- The security manager does not ordinarily allow applets to execute arbitrary system commands, to load system libraries, or to open up system device drivers such as disk drives. In addition, scripts are generally limited to reading and writing to files in a user-designated directory only.

- Applets are also limited in the network connections they can make: An applet is only allowed to make a network connection back to the server from which it was downloaded.

- Finally, the security manager allows Java applets to read and write to the network, read and write to the local disk, but not both. This limitation was created to reduce the risk of an applet spying on the user's private documents and transmitting the information back to the server.

Unfortunately in the short time since its release, a number of security holes have been found in Java caused by bugs in the implementation. A variety of bugs have been identified and fixed.

Below are some of the older security holes present in the Java implementations distributed with various versions of Netscape and Internet Explorer.

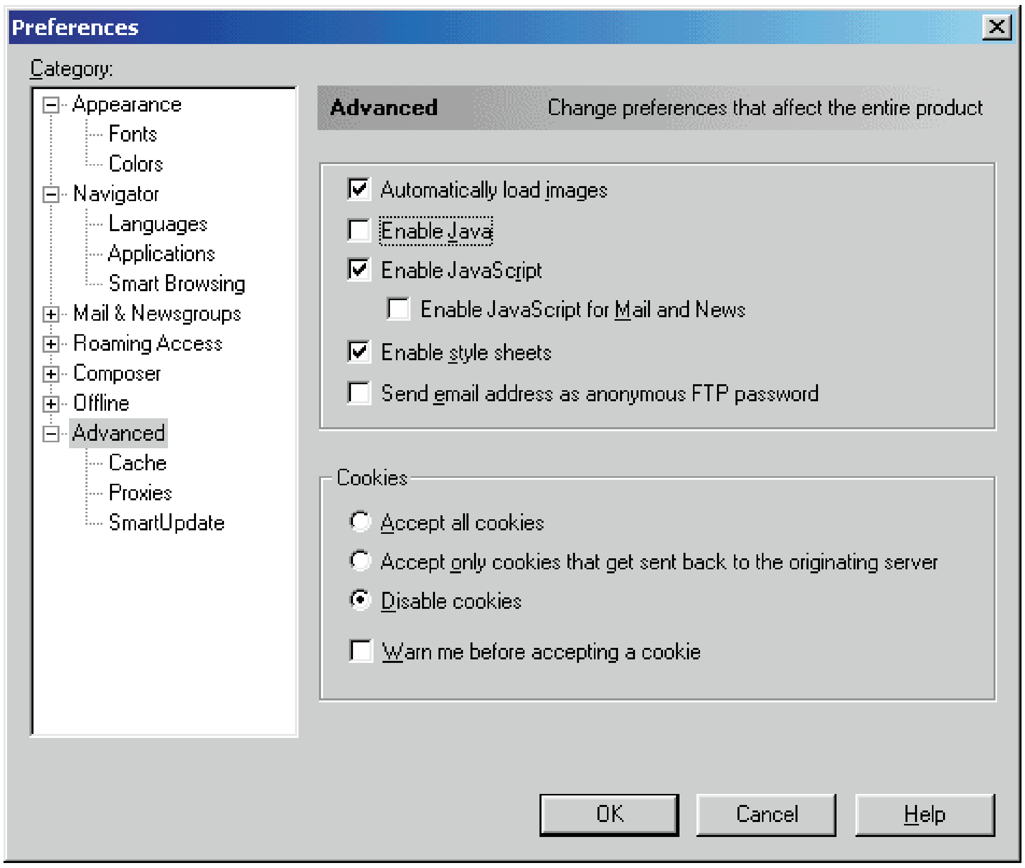
- Vulnerability to denial-of-service attacks:
  Applets can hog system resources such as memory and CPU time. This may happen as the result of a programmer error, or maliciously in order to slow down the computer system to the point of unusability. Applets running under the same browser are not protected from one another. One applet can easily discover another's existence and interfere with it, raising the interesting spectre of one vendor's applet deliberately making a competitor's applet appear to behave erroneously.

- Ability to make network connections with arbitrary hosts:
  Once downloaded to a user's machine, the applet can attempt to make a connection to any machine on the user's local area network, even if the LAN is protected by a firewall. Many LANs are set up so that local machines are trusted to access services that distant machines are not. As a trivial example, an Applet could open up a connection to the organization's private news server, fetch recent postings to an internal newsgroup, and transmit them back to a foreign host.

If you are security conscious, you might wish to take the safest course and deactivate Java completely. In Netscape Navigator 2.0-3.02, you can do this by unchecking the "Java" option in **Edit->Preferences->Advanced**. In Internet Explorer 3.02, uncheck `Enable Java Programs` in the **View->- Options->Security->Active Content** window.

Deactivating Java is harder in the 4.0 versions of both Navigator and Internet Explorer. In Netscape Navigator 4.0, select **Edit->Preferences** from the menu bar, then select the "Advanced" category. Locate the "Enable Java" checkbox and deselect it.

In IE 4.0, select **View->Internet Options->Security**, select the Internet Zone, and select `Custom` settings. Now press the `Settings...` button and scroll down to the Java settings. Choose `Disable Java`.



Advanced Change preferences that affect the entire product

[✓] Automatically load images
[ ] Enable Java
[✓] Enable JavaScript
[ ] Enable JavaScript for Mail and News
[✓] Enable style sheets
[ ] Send email address as anonymous FTP password

Cookies
( ) Accept all cookies
( ) Accept only cookies that get sent back to the originating server
(●) Disable cookies
[ ] Warn me before accepting a cookie

Fig. 4.4-2: How to disable Java in Netscape Navigator 4.0 or higher.

4.4.4 ActiveX

ActiveX is a technology developed by the Microsoft Corporation for distributing software over the Internet. It is an outgrowth of two other Microsoft technologies called OLE (Object Linking and Embedding) and COM (Component Object Model). Like Java Applets, an ActiveX `control` can be embedded in a Web page, where it typically appears as a smart interactive graphic. A number of ActiveX controls are available for the Microsoft Internet Explorer (the only browser to support them so far), including a control that executes Java applets. Unlike Java, which is a platform-independent programming language, ActiveX controls are distributed as executable binaries, and must be separately compiled for each target machine and operating system.

Security model

The ActiveX security model is considerably different from Java applets. Java achieves security by restricting the behavior of applets to a set of safe actions. ActiveX, on the other hand, places no restrictions on what a control can do. Instead, each ActiveX control can be digitally signed by its author in such a way that the signature cannot be altered or repudiated using a system called `authenticode` (a special kind of certificate). The digital signatures are then certified by a trusted `certifying authority`, such as VeriSign, to create the equivalent of a shrink-wrapped software package. When a digital certificate is granted, the software developer pledges that the software is free from viruses and other malicious components.

This security model places the responsibility for the computer system's security squarely on the user's side. Before the browser downloads an ActiveX control that hasn't been signed at all, or that has been signed but certified by an unknown certifying authority, the browser presents a dialog box warning the user that this action may not be safe. The user can select to abort the transfer, or may continue the transfer and take his chances.

The ActiveX certification process ensures that ActiveX controls cannot be distributed anonymously and that a control cannot be tampered with by third parties after its publication. However, the certification process does not ensure that a control will be well-behaved. Although it is unlikely that signed and certified ActiveX controls will behave in a malicious fashion, this is not impossible.

The main problem with the ActiveX security model is that it is difficult to track down a control that has taken some subtle action, such as silently transmitting confidential configuration information from the user's computer to a server on the Internet, seeding the LAN with a virus, or even patching Internet Explorer so that the code authentication engine no longer functions correctly. This type of action may escape detection completely, or at least for a long period of time. Even if the damage is detected immediately, Internet Explorer offers no secure audit trail that records which ActiveX controls were downloaded. This makes identifying the control responsible for damaging your system a non-trivial task.

ActiveX can be turned off completely from the **Internet Options->Security** pages of Microsoft Internet Explorer. The default setting is on. This means that users who are unaware of security considerations will automatically run any ActiveX scripts that their browser downloads. In addition, there is an option to set the safety level. Choose the `High Security` setting to disable ActiveX completely, or `Medium Security` to prompt you before downloading and executing ActiveX controls. The `Low Security` option allows any ActiveX control to run, signed or not, and is not recommended.

4.4.5 Cookies

We will take a look at the basic technology behind cookies, as well as some of the features they enable. A cookie is a piece of text that a web server can store on a user's hard disk. Cookies allow a web site to store information on a user's machine and later retrieve it. The pieces of information are stored as name-value pairs. For example, a web site might generate a unique ID number for each visitor and store the ID number on each user's machine using a cookie file.

If you use Microsoft's Internet Explorer to browse the web, you can see all of the cookies that are stored on your machine. The most common place for them to reside is in a directory called

```
c:\windows\cookies
```

You can see in the directory that each of these files is a simple, normal text file. You can see which web site placed the file on your machine by looking at the file name (the information is also stored inside the file). You can open each file up by clicking on it.

For example, a cookie file created by `amazon.com` may contain the following information:

```
session-id 104-1889894-2550358 amazon.com/
session-id-time 992851200 amazon.com/
ubid-main 077-0190749-3709979 amazon.com/
x-main hQFiIxHUFj8mCscT@Yb5Z7xsVsOFQjBf amazon.com/
```

It appears that Amazon stores a main user ID, an ID for each session, and the time the session started on the user's computer (as well as an `x-main value`, which could be anything).

The vast majority of sites store just one piece of information - a user ID - on your computer. But there really is no limit - a site can store as many name-value pairs as it likes. A name-value pair is simply a named piece of data. It is not a programme, and it cannot ”do” anything. A web site can retrieve only the information that it has placed on your machine. It cannot retrieve information from other cookie files, nor any other information from your machine.

How does cookie data move?

As shown in the previous section, cookie data is simply a name-value pair stored on your hard disk by a web site. That is all that cookie data is. The web site can store the data, and later it receives it back. A web site can only receive the data it has stored on your computer. It cannot look at any other cookie, nor can it look at anything else on your computer. The data moves in the following manner:

- If you type the URL of a web site into your browser, your browser sends a request to the web site for the page. For example, if you type the URL http://www.amazon.com into your browser, your browser will contact Amazon's server and request its home page.
- When the browser does this, it will look on your machine for a cookie file that Amazon has set. If it finds an Amazon cookie file, your browser will send all of the name-value pairs in the file to Amazon's server along with the URL. If it finds no cookie file, it will send no cookie data.
- Amazon's web server receives the cookie data and the request for a page. If name-value pairs are received, Amazon can use them.
- If no name-value pairs are received, Amazon knows that you have not visited their site before. The server creates a new ID for you in Amazon's database and then sends name-value pairs to your machine in the header for the web page it sends. Your machine stores the namevalue pairs on your hard disk.
- The web server can change name-value pairs or add new pairs whenever you visit the site and request a page.

There are other pieces of information that the server can send with the name-value pair. One of these is an expiration date. Another is a path (so that the site can associate different cookie values with different parts of the site). You have control over this process. You can set an option in your browser so that the browser informs you every time a site sends name-value pairs to you. You can then accept or deny the values.

How do web sites use cookies?

Cookies evolved because they solve a big problem for people who implement web sites. In the broadest sense, a cookie allows a site to store state information on your machine. This information lets a web site remember what state your browser is in. An ID is one simple piece of state information - if an ID exists on your machine, the site knows that you have visited it before. The state is, ”Your browser has visited the site at least one time”, and the site knows your ID from that visit. Web sites use cookies in many different ways. Here are some of the most common examples:

- Sites can accurately determine how many readers actually visit the site. It turns out that because of proxy servers, caching, concentrators and so on, the only way for a site to accurately count visitors is to set a cookie with a unique ID for each visitor. Using cookies, a site can:

 – Determine how many visitors arrive
 – Determine how many are new vs. repeat visitors
 – Determine how often a visitor has visited.

 The way the site does this is by using a database. The first time a visitor arrives, the site creates a new ID in the database and sends the ID as a cookie. The next time the user comes back, the site can increment a counter associated with that ID in the database and know how many times that visitor returns.

- Sites can store user preferences so that the site can look different for each visitor (often referred to as customisation). For example, if you visit msn.com, it offers you the ability to change `content/layout/color`. It also allows you to enter your zip code and get customized weather information. Most sites seem to store preferences like this in the site's database and store nothing but an ID as a cookie, but storing the actual values in name-value pairs is another way to do it (we'll discuss why this approach has lost flavour below).
- eCommerce Sites can implement things like shopping carts and ”quick check-out” options. The cookie contains an ID and lets the site keep track of you as you add different things to your cart. Each item you add to your shopping cart is stored in the site's database along with your ID value. When you check out, the site knows what is in your cart by retrieving all of your selections from the database. It would be impossible to implement a convenient shopping mechanism without cookies or something like it.

In all of these examples, note that what the database is able to store is things you have selected from the site, pages you have viewed from the site, information you give to the site in online forms, etc. All of the information is stored in the site's database, and a cookie containing your unique ID is all that is stored on your computer in most cases.

Privacy

On a web site, the site can track not only your purchases, but also the pages that you read, the ads that you click on, etc. If you then purchase something and enter your name and address, the site potentially knows much more about you than a traditional mail order company does. This makes targeting much more precise, and that makes a lot of people uncomfortable.

The second fact is new. There are certain infrastructure providers that can actually create cookies that are visible on multiple sites. DoubleClick is the most well known example of this. Many companies use DoubleClick to serve ad banners on their sites. DoubleClick can place small (1x1 pixels) GIF files on the site that allow DoubleClick to load cookies on your machine. DoubleClick can then track your movements across multiple sites. It can potentially see the search strings that you type into search engines. Because it can gather so much information about you from multiple sites, DoubleClick can construct very rich profiles. These are still anonymous, but they are rich on information.

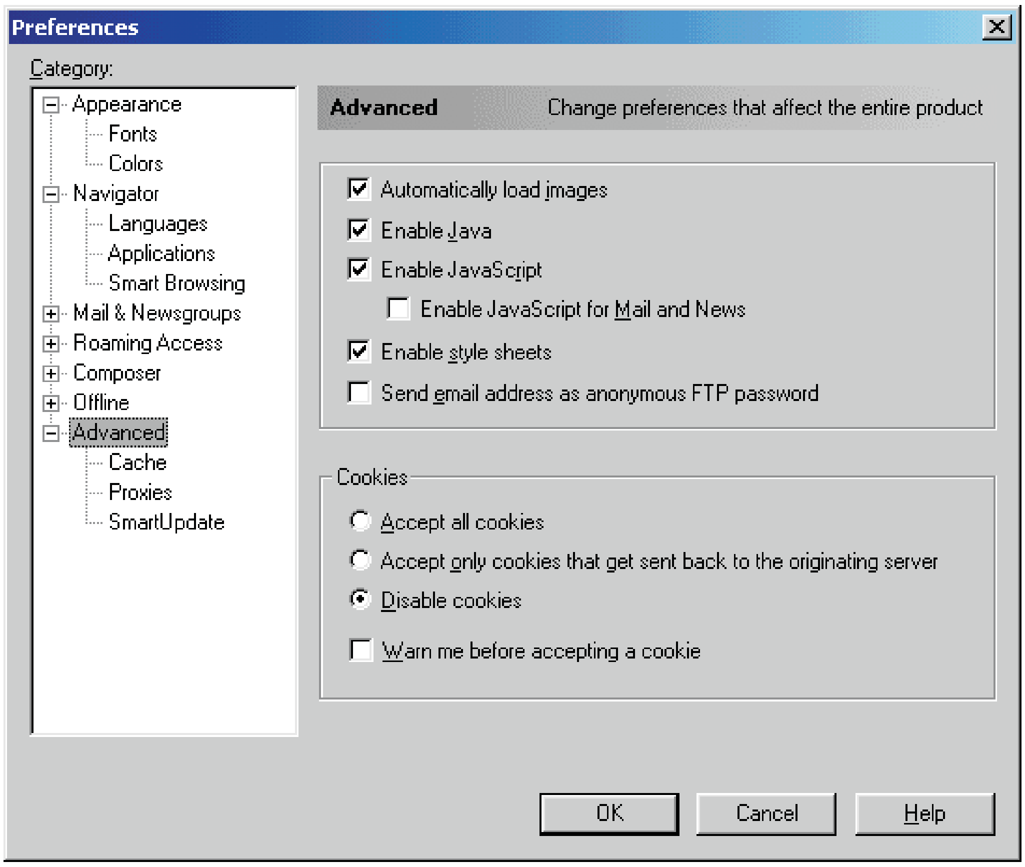
DoubleClick then went one step further. By acquiring a company, DoubleClick threatened to link these rich anonymous profiles back to name and address information – it threatened to personalize them, and then sell the data. That began to look very much like spying to most people, and that is what caused uproar.

DoubleClick and companies like it are in a unique position to do this sort of thing, because they serve ads on many sites. Cross-site profiling is not a capability available to individual sites, because cookies are site specific.

How to disable cookies?

Current versions of both Netscape Navigator and Internet Explorer offer the option of alerting you whenever a server attempts to give your browser a cookie. If you turn this alert on, you will have the option of refusing cookies. You should also manually delete any cookies that you have already collected. The easiest way to do this is to remove the cookies file entirely.

The drawback to this scheme is that many servers will offer the same cookie repeatedly even after you refuse to accept the first one. This rapidly leads to a nuisance situation. Netscape Navigator 4.0 or higher provide a new feature that allows you to refuse cookies that are issued from sites other than the main page you are viewing. This foils most DoubleClick schemes without interfering with the more benign cookies. To access this option, select **Edit->Preferences->Advanced**, and select the appropriate radio button from the cookies section (see Fig. 4.4-3). Some people might want to allow transient cookies (ones active only during a browsing session) but forbid persistent ones (ones that store user identification information over an extended period). On Unix systems, you can do this easily by creating a symbolic link between the Unix device `/dev/null` and the cookies file. Users of other operating systems may have to invest in third party products that intercept cookies.



Advanced Change preferences that affect the entire product

- [✓] Automatically load images
- [✓] Enable Java
- [✓] Enable JavaScript
  - [ ] Enable JavaScript for Mail and News
- [✓] Enable style sheets
- [ ] Send email address as anonymous FTP password

Cookies

- ○ Accept all cookies
- ○ Accept only cookies that get sent back to the originating server
- ● Disable cookies
- [ ] Warn me before accepting a cookie

Fig. 4.4-3: How to disable cookies in Netscape Navigator 4.0 or higher.

Exercise 4.4-1:

Specify the types of security risks regarding the WWW.

Exercise 4.4-2:

Shortly describe the two methods used in HTTP for user authentication.

5 Anonymity Techniques

5.1 Introduction

It is desirable in some communication sessions to conceal the identity of an entity involved in this communication. That is, a participant wants anonymously to use communication services without compromising his identity. For example, one wants to send, and/or receive messages in an electronic mail system without being observed. In this approach Chaum discussed the **"traffic analysis problem"**

Traffic analysis problem

"The problem of keeping confidential who communicates with whom; and when they communicate"

One important concept in anonymity techniques is the **Mix-concept**. Its basic idea was published by Chaum in 1981 ([Chaum81]). Thereafter, it was adjusted in order to guarantee anonymity in some communication networks such as mobile communication networks or telephony communication networks. Anonymity can also be offered by another concept: the **DC-concept** or DC network.

Mix-concept

DC-concept

This chapter deals with some basic ideas of anonymity used in the Mix- and the DC-concept and introduces the following types of anonymity:

Sender anonymity The sender of a message stays anonym for the recipient as well as for an observing attacker.

Sender anonymity

Recipient anonymity In a communication relationship between two partners A and B **recipient anonymity** allows e.g. A to send a message to B without breaking the anonymity of B. For this purpose A becomes a special address which refers to B but does not reveal $B's$ identity to A.

Recipient anonymity

Mutual anonymity Combines sender and recipient anonymity and extends the concepts to provide anonymity for both, sender and recipient.

Mutual anonymity

Unobservability The communication between two partners A and B can not be observed, not even the existence of a communikation link between A and B can be revealed by a third party.

Unobservability

For client-server-applications like the internet service WWW, we introduce the notion offered by the JANUS system:

Client anonymity **Client anonymity** is reached, when all information about the client is concealed.

Client anonymity

Server anonymity **Server anonymity** is reached, when the identity of a server in a communication session is hidden.

Server anonymity

5.2 The Mix-concept

A mix is a network node that enables to hide the relation between communicating parties and to guarantee sender anonymity, recipient anonymity, mutual anonymity or the unlikability of sender and recipient as partners of the communication. It is a solution, proposed by Chaum, to the traffic analysis problem. Using public key cryptography, the purpose of a mix is to keep confidential who communicates with whom and when they communicate.

For simplicity and without loss of generality, we observe the communication in a mail system between two parties A (Alice) and B (Bob) through a mix M. Each entity A and B, and the mix M possesses the pair of public and private keys $(k_{e,A}, k_{d,A})$, $(k_{e,B}, k_{d,B})$, and $(k_{e,M}, k_{d,M})$ respectively. We suppose that A wants to send a message m to B, then she proceeds as follows:

A appends random bits R_0 to the message m, encrypts all with B's public key and obtains

$$E_{k_{e,B}}(m, R_0) = c_1.$$

Thereafter, she encrypts the destination address B, choses random bits R_1 and c_1 with M's public key $k_{e,M}$ and transmits

$$E_{k_{e,M}}(R_1, c_1, B) = c_2$$

to M. The mix M decrypts c_2 with his own private key $k_{d,M}$ and obtains

$$D_{k_{d,M}}(c_2) = (R_1, c_1, B).$$

Then, M takes out the random bits R_1 from (R_1, c_1, B), determines the next destination address B and transmits the message $c_1 = E_{k_{e,B}}(m, R_0)$ to it (Fig. 5.2-1).

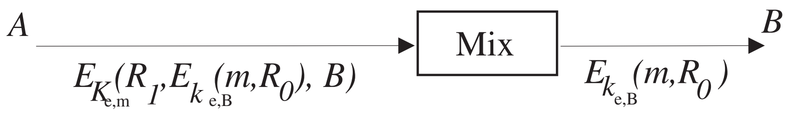


Fig. 5.2-1: Principle of a mix.

The encryption and decryption performed by a mix permit to change appearances of the incoming messages. Finally B uses his private key $k_{d,B}$ to decrypt c_1. B performs the operation

$$D_{k_{d,B}}(c_1) = (m, R_0)$$

and gets the message m after removing the random bits R_0.

Of course, if only one message has been sent through the mix M, then the relationship between the sole incoming and outcoming message would be obvious. This requireds that a number of messages m_1, m_2, m_3, $\ldots$ called a **batch** must be first collected and then treated (de/- encrypted) at one time. An attacker can thus not find a **time correlation** between incoming and outcoming messages. In order to avoid a **replay-attack**, each message must be treated by the mix no more than once. Repeated messages in a batch must also be discard before being processed by the mix. In other words, if copies of a message exist in a batch, then the message can be traced by observing the **frequency of occurrence** between the incoming and the outcoming messages. A mix must also be able to remove all repeated messages when they appear in multiple batches.

Batch

Time correlation

Replay-attack

Frequency of occurrence

Which role do the random bits R_0 and R_1 play? Supposing that a message m is encrypted by a mix without attachment of any random bits, then an attacker can simply re-encrypt the outcoming message $D_{k_{d,M}}(E_{k_{e,M}}(m))$ with the mix's public key and obtain

$$E_{k_{e,M}}(D_{k_{d,M}}(E_{k_{e,M}}(m))) = E_{k_{e,M}}(m).$$

Now he transmits anew the encrypted message $E_{k_{e,M}}(m)$ to the mix M which decrypts it with his private key and outputs the message m. Hence a correspondence between the incoming and the outcoming message is found and the relation between sender and recipient of a message is not hidden anymore.

Another operation performed by the mix is the **reordering** of all messages in a batch. All incoming messages of a batch are reordered in a lexicographical order and then outputted. This prevents an attacker to detect the correspondences between sequential inputs and their sequential outputs.

Reordering

Despite all operations described until now which can be performed by the mix in order to hide any relation between the incoming and the outcoming messages, an attacker can possibly find out which outgoing message corresponds to which incoming one only by observing the length of each message. Therefore all incoming and outgoing messages must be of equal length. For this purpose, the messages should be padded if necessary .

Fig. 5.2-2 resumes some basic functions of a mix. In practice, other supplementary functions of a mix can be needed in order –for example– to handle latency, time delay of information in a network etc or to avoid some known attacks against a mix (see [Franz97] and [Franz98]).

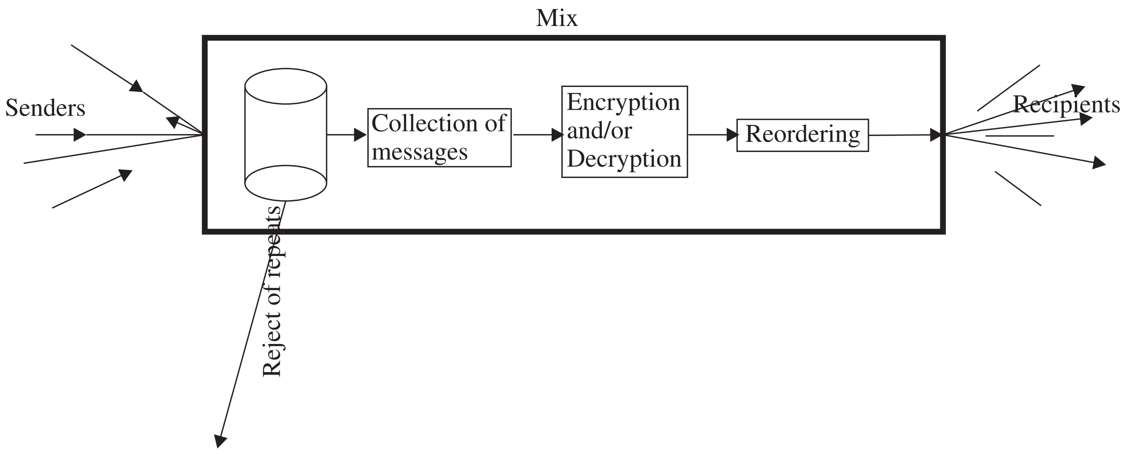


Fig. 5.2-2: Some basic functions of a mix.

If only one mix is used, we can determine the source and the destination of a message whereas the relation between sender and recipient of a message is hidden from everyone except the mix and the sender of a message ([Pfitzmann87]). In other words, one mix – if it is trustworthy – guarantees only the unlinkability of sender and recipient. In order to attain more functionalities, e.g. sender anonymity, we use a sequence of mixes where at least one mix must be trustworthy. In a serie of mixes M_i, $i = 1, 2, \ldots, n$ (see Fig. 5.2-3) each mix M_i performs all the operations, described above. Each M_i collects a batch of messages, discards repeats, carries out cryptographic operations to each message of a batch and outputs a batch in a lexicographical order.

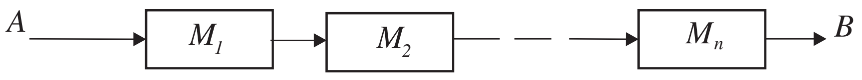


Fig. 5.2-3: Sequence of mixes between two communication partners A and B

5.2.1 Sender anonymity

A wants to send a message m to B in such a way that B receives m but without knowing who sends the message. So A determines a sequence of mixes M_i, $i = 1$, 2, ..., n. Each mix M_i possesses a pair of public and a private key (k_{e,M_i}, k_{d,M_i}). Then she prepares the message by encrypting it in the following manner

$$m_1 = E_{k_{e,B}}(R_0, m, B)$$
$$m_i = E_{k_{e,M_{i-1}}}(R_{i-1}, m_{i-1}, A_{i-2}) \text{ for } i = 2, \ldots, n+1.$$

A sends the message

$$m_{n+1} = E_{k_{e,M_n}}(R_n, m_n, A_{n-1})$$

to the mix M_n which decrypts it with his private key k_{d,M_n} and obtains (R_n, m_n, A_{n-1}). Thereafter, M_n removes the random bits R_n and determines the next destination address A_{n-1} of the mix M_{n-1} to which it sends the message m_n. The mixM_{n-1} proceeds in the same manner as M_n. In this way all mixes will be successively traversed (see Fig. 5.2-4). The mix M_1 finaly gets the message m_2 and sends the message m_1 to the destination address B. Hence B receives the message m without knowing the identity of the sender.

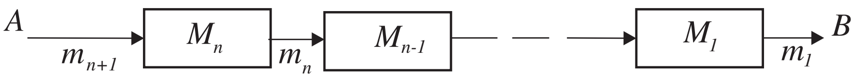


Fig. 5.2-4: Sender anonymity in a mix network.

Sender anonymity can be seen as packing the message m in n nested envelopes. Beginning with the outmost envelope n and proceeding in descending order, the address A_i of the mix M_i is written down on each envelope i. So the outmost envelope can only be removed by the mixM_n, the next envelope by the mix M_{n-1}, etc. This is the same as the decryption with the mixes' private keys. The last mix M_1 forwards the intended message m to the recipient B in an extra envelope that can only be opened by B .

5.2.2 Return address and recipient anonymity

Anonymous return address

B wants to receive a message m anonymously from A. Therefore he must prepare an address called **anonymous return address** (RA) which referes to be but does not reveal his identity.

Again we consider a sequence of mixes M_1, M_2, $\ldots M_n$ which is choosen by B to assign the way message m should take through the Mix network. Each mix M_i possesses a pair of public and private keys (k_{e,M_i}, k_{d,M_i}) and has the address A_i. B generates RA by passing the sequence of mixes in descending order of the mix indices. For each M_i in the sequence a part R_i of the return address will be formed as

$$R_i = E_{k_{e,M_i}}(k_i, A_{i+1}, R_{i+1}), \text{ for } i = n, \ldots 1$$

where k_i is a symmetric key, associated with the mix M_i, that acts as random string ([Chaum81]). In particular, A_{n+1} denotes the address of B and $R_{n+1} = r$ is a stamp for B. We notice here that each mix must not treat repeats of return address parts. Otherwise an attacker will be able to trace the message passing through such a mix.

With the notation introduced above, the return address RA should be equal to (k_0, A_1, R_1), where k_0 is a symmetric key associated with the sender. After preparing the address RA, B sends it anonymously to A (see Fig. 5.2-5) using a sender anonymity scheme.

Bemerkung: B can choose differnt ways for m and RA through the Mix network.

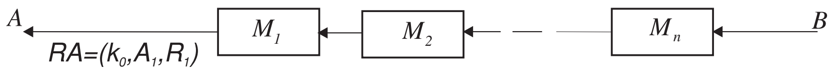


Fig. 5.2-5: Recipient anonymity in a mix network: Return address preparation.

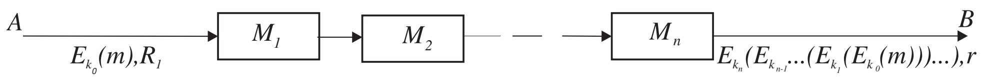


Fig. 5.2-6: Recipient anonymity in a mix network: Anonymous sending of m.

Now A uses the prepared return address to send the message m. So she encrypts the message with the key k_0 and sends R_1, $E_{k_0}(m)$ to the mix M_1. Since

$$R_1 = E_{k_{e,M_1}}(k_1, A_2, R_2),$$

the mix M_1 can decrypt R_1 with his private key k_{d,M_1} and obtain k_1, A_2, R_2. Then he encrypts $E_{k_0}(m)$ with k_1 as

$$E_{k_1}(E_{k_0}(m))$$

and sends the message R_2, $E_{k_1}(E_{k_0}(m))$ to the mix M_2. Proceeding in this manner, B gets the message

$$(R_{n+1} = r), \quad E_{k_n}(E_{k_{n-1}} \ldots (E_{k_1}(E_{k_0}(m)))).$$

B can also decrypt $E_{k_n}(E_{k_{n-1}} \ldots (E_{k_1}(E_{k_0}(m))))$, since the order of the used encryptions is known to him (Fig. 5.2-6).

5.2.3 Mutual anonymity

By combining both sender and recipient anonymity, we get mutual anonymity. For this purpose, one mix M_R plays the role of the *turning point* for sender and recipient. So the recipient generates his return address as M_R to which the sender sends anonymously a message.

The recipient can trace the message only up to the mix M_R, whereas the recipient cannot control the message after passing through M_R.

5.2.4 Anonymity services

The Internet is the global association of computers carrying data and making the exchange of information possible. The World Wide Web (WWW) is a service of the Net. It is a collection of interlinked documents that work together using a specific Internet protocol called HTTP (HyperText Transfer Protocol). This protocol is also used to transmit and receive all data over the World Wide Web. When we type a URL[27] into our browser, we are actually sending an HTTP request to a Web server for a page of information. Simultaneously, we send unintentionaly lot of administrative data such as the E-mail address of our browser, the version of our operating system ...etc. On the other side, the server gets the HTTP request and reacts with a HTTP response, i.e. it sends the desired page of information to the client and some additional administrative data. The underlying communication utilized here is accorded to the client/server model. The anonymity of the communication based on our previous model can be specified in terms of client anonymity and/or sender anonymity ([Demuth01]).

Client anonymity

In a communication based on the client/server model, client anonymity is reached when all information about the client is concealed, i.e. the client is kept anonymous during a communication session ([Demuth01]). There exist some approaches that guarantee client anonymity. For example, in the**proxy**[28] **approach**, the proxy

Proxy approach

27 The URL (Uniform Resource Locator) is the Internet equivalent of an address. An example of a URL is http://www.fernuni-hagen.de.

receives an HTTP request and removes or modifies all sensitive information of the client before forwarding the HTTP request to the corresponding web server. In this case, the proxy can be considered as an intermediary party preventing unauthorized parties[29] from gathering personal information about the client's usage. There are already several services on the web which enable client anonymity. The Anonymizer is one of such services, offered in the Internet, that provides client anonymity using the proxy approach.

Server anonymity

The server anonymity enables to hide the identity of a server in a communication session. In fact, services that provide server anonymity, are rare in comparison to that which offer client anonymity. The JANUS (justly anonymising numerous URLs systematically) system is a service which supports both client and server anonymity in the WWW. It represents a network of stations working with the mix method. In the JANUS system, the communication between client and server can occur over one JANUS instance or more instances in order to increase the security. In Fig. 5.2-7, we see that a communication between client 3 and server 2 occurs over one JANUS instance (JANUS 3).

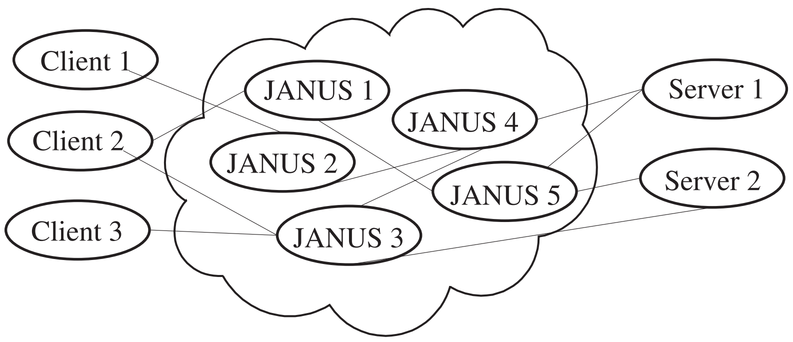


Fig. 5.2-7: JANUS network.

Each JANUS instance has a pair of public and private keys. When a provider wishes to publish an information on a web site without revealing his identity (server anonymity), he has to encrypt the URL with the public key of a JANUS instance or optionally with more than one public key of many JANUS instances. The encryption occurs using public keys published by the operators of the JANUS server or simply by accessing a special web page provided by a JANUS-web-server. The encrypted URL is used as a return address as in the mix concept. By using the JANUS system, the URL ([Demuth98])

28 Here, proxy means the proxy server which is a programm acting as an intermediary between a web browser and a web server. A proxy server is mostly used to store frequently demanded pages. Thus it gives users rapid access to popular web destinations without overloading the network with uncessary communications traffic.

29 for example, the provider of the web server.

http://ks.fernuni-hagen.de/

is encrypted into

http://www.rewebber.de/surf_encrypted/MTBLKVGnZFbdGsfN2AaN8CRp7JFT
$djND1CktOFNdNq8ow$eFW5Hq+J4ubGlxK+ob4ocVDScx$3YCf$N6cVxd9bK
18h4iG93cPuvb3q5quijGz$SeCYq+OkqIVESlTsccQo=

Thereafter, the encrypted URL can be published in such a way that the anonymity of the provider remains guaranteed. In this manner, an observer of the encrypted URL can not gain any information about the provider of the web site. Of course, the content of the provided web site should not contain any indications regarding the provider such as his explicit address. Otherwise the anonymity will be clearly violated.

Now each user that types the encrypted URL in his browser sends an HTTP request to the corresponding JANUS instance[30]. The last instance decrypts the encrypted URL with his private key and removes or modifies all sensitive informations related to the client. In other words, the JANUS system guarantees also client anonymity. Thereafter an HTTP request is forwarded by the JANUS instance to the corresponding web server which delivers the desired web site. The JANUS instance analyses again the received page (from the web server) for URLs to be encrypted and for sensitive information (of the web server) to be modified or removed, before forwarding this page to the client.

5.3 DC-network

The concept of a DC-network was introduced by Chaum in his publication titled *"the **D**ining **C**ryptographers problem: Unconditional sender and recipient untraceability"*([Chaum88]):

> *"Three cryptographers are sitting down to dinner at their favorite three-star restaurant. Their waiter informs them that arrangements have been made with the maitre d' hotel for the bill to be paid anonymously. One of the cryptographers might be paying for the dinner, or it might have been NSA(U.S. National Security Agency). The three cryptographers respect each other's right to make an anonymous payment, but they wonder if NSA is paying."*

To resolve this problem, Chaum proposes that each cryptographer flip a coin[31] in secrecy between him and his right neighbour. Each of the three cryptographers must also state whether the faces seen at his right and his left are the same or not. If one cryptographer has arranged to pay, he says the opposite of what he sees. Hence an odd number of cryptographers saying "different" implies that one of them paid,

30 We assume here that the URL was encrypted from only one JANUS instance, the other case can be treated in the same manner.

31 each face of the coin (P/F) is required to appear with the probability ½.

an even number indicates that NSA is paying. The proposed protocol is performed under the assumption that the dinner is to be paid by one person only.

Example: 5.1
Let A, B, and C[32] be three cryptographers. Without loss of generality, supposing that A says "different" and A does not tell the opposite of what she sees. For example, A sees the face P with B and the face F with C (the other case, where A sees the face F with B and the face P with C can be treated in the same manner as in this example). Can A determine whether B or C has paid? Two subcases could occurre: either the face, seen between B and C, is P or F (see Fig. 5.3-1).

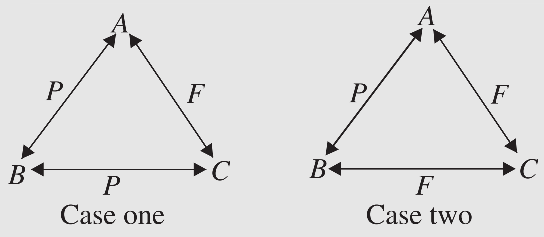


Fig. 5.3-1: Cases that can occur if the face seen between A and B resp. C is P resp. F

1. The face, seen between B and C, is P: In this case A obtains the following table[33]

Tab. 5.3-1: *Case one in the three cryptographers example.*

A	B	C	Number of d	who does tell the opposite	The payer
d	s	d	2	-	NSA
d	d	d	3	B	B
d	s	s	1	C	C
d	d	s	2	B, C	Not possible

d resp. s means that the corresponding cryptographer says "different" resp. "same".

2. The face seen between B and C is F: In this case A obtains the following table

Tab. 5.3-2: *Case two in the three cryptographers example.*

A	B	C	Number of d	who does tell the opposite	The payer
d	s	d	2	B, C	Not possible
d	d	d	3	C	C
d	s	s	1	B	B
d	d	s	2	-	NSA

From the two tables we conclude: If the number of differences is even (=2), then A know immediately that NSA has paid. Under the assumption that the dinner can be paid for only once, two (three) cryptographers cannot simultaneously tell the opposite of what they see. For this reason the last (the first) row in Table 5.3-1 (Table 5.3-2) is not possible. The cases one and two can occur with equal probability. That is, if an odd number of cryptographers say "different", then A knows that the dinner is paid but she has no information who is the payer.

If we proceed as in the above Example and analyze all cases that can occur between three cryptographers, we conclude that the anonymity of each cryptographer is guaranteed. If one cryptographer who is not the payer tries to find out the payer (if the dinner is not paid by NSA), he gets no information. Hence the protocol proposed by Chaum in order to resolve the "dining cryptographers problem" is **unconditionally secure** (information-theoretically secure).

Unconditionally secure

The idea of Chaum can be mpdified in order to provide an anonym sending of messages in a network. Supposing that each user station can send a message of only one bit. So each user generates a one bit key and sends it to one other user over a secure channel. Subsequently each user has 2 key bits: one he generates and one he receives. If a user wants to send a one message bit, then he inverts the two key bits he possesses, adds up the inverses modulo 2 and the one message bit he wishes to send. The user, which has no message to send, adds up modulo 2 his two keys (one he generates and one he receives). The sum delivers the one message bit that will be distributed over all users. Note that the inverse of an element in GF(2) is the element itself.

Example: 5.2
Let A, B and C three users of a network. A, B and C resp. generate a key bit k_a, k_b, k_c and send it to B, C and A resp. Then A, B and C receive the key bit k_c, k_a and k_b. A wants to send a message bit m, then she proceeds as follows:

Each users A, B and C resp. forms the sum $s_a = m + k_a + k_c$, $s_b = k_b + k_a$ and $s_c = k_c + k_b$ resp. The superposition of all sums delivers

$$\begin{aligned} s_a + s_b + s_c &= m + k_a + k_c + k_b + k_a + k_c + k_b \\ &= m + \underbrace{k_a + k_a}_{=0} + \underbrace{k_b + k_b}_{=0} + + \underbrace{k_c + k_c}_{=0}. \\ &= m. \end{aligned}$$

This result will be distributed over all users. Each user receives the message bit but cannot find out the sender.

32 Carolin.

33 Under the assumption that the dinner can be paid for only once.

Generally, the protocol of Chaum can be applied in a network of many user stations and for sending messages of many bits. So each user, that wants to send a message, must proceed for each message bit as shown above. This will be illustrated in the following example.

Example: 5.3
We use the same notation as in Example. Let j be an integer ≥ 1 and $m = (m_1, m_2, \ldots, m_j)$. Let

$$\begin{aligned} k_a &= (k_{a,1}, k_{a,2}, \ldots, k_{a,j}) \\ k_b &= (k_{b,1}, k_{b,2}, \ldots, k_{b,j}) \\ k_c &= (k_{c,1}, k_{c,2}, \ldots, k_{c,j}) \\ k_{n,m} &\in \{0, 1\}. \end{aligned}$$

A wishes to send the message m. Then she forms the sum $s_a = (s_{a,1}, s_{a,2}, \ldots, s_{a,j})$ such that for $1 \leq i \leq j$,

$$s_{a,i} = m_i + k_{a,i} + k_{c,i} \mod 2.$$

B does not want to send, he forms the sum s_b such that for $1 \leq i \leq j$,

$$s_{b,i} = k_{b,i} + k_{a,i} \mod 2.$$

C has no message to send, she forms the sum s_c such that for $1 \leq i \leq j$,

$$s_{c,i} = k_{c,i} + k_{b,i} \mod 2.$$

The superposition of all sums delivers

$$\begin{aligned} S_i &= s_{a,i} + s_{b,i} + s_{c,i} \mod 2. \\ &= m_i + k_{a,i} + k_{c,i} + k_{b,i} + k_{a,i} + k_{c,i} + k_{b,i} \mod 2. \\ &= m_i + \underbrace{k_{a,i} + k_{a,i}}_{=0} + \underbrace{k_{b,i} + k_{b,i}}_{=0} + \underbrace{k_{c,i} + k_{c,i}}_{=0} \mod 2. \\ &= m_i, \quad \text{for } 1 \leq i \leq j. \end{aligned}$$

The result is the message $m = (m_1, m_2, \ldots, m_j)$ that will be distributed over all users. Fig. 5.3-2 illustrates this example with a key length of four bits.

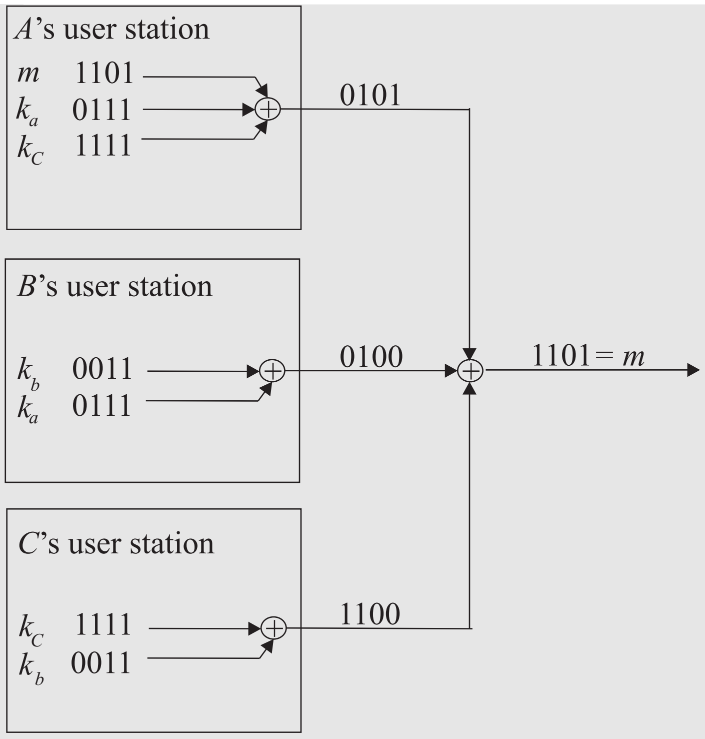


Fig. 5.3-2: Sender anonymity in a DC-network.

In fact, it is usually the case that more than one user station tries to send simultaneously. In this case, the users will notice that the overall outcome does not coincide with the message each one desired to send (in Example$(S_1, S_2, \ldots S_j) \neq (m_1, m_2, \ldots m_j)$. This phenomenon is called a **collision** and is typical in a network. If the users detect a collision, then they must stop to send and wait a random number of rounds before retransmitting again. There exist some multi-access methods which regulate the message traffic in a network. An adequately handling of these methods is beyond the scope of this course.

Collision

One network topology which can be adapted in order to carry out the protocol of Chaum is the **ring topology**. It is in wide use for local area networks. In the ring network, the users are connected on a cycle.

Ring topology

"Consider a single-bit message like the "I paid" message originally sent at the dinner table. Each participant exclusive-or's the bit he receives with his own output before forwarding it to the next participant. When the bit has traveled a full circle, it is the exclusive-or sum of all the participants' outputs, which is the desired result of the protocol. To provide these messages to all participants, each bit is sent around a second time by the participant at the end of the loop"([Chaum88]).

As mentioned in this section, all sent messages in a DC-network will be delivered to each participant in the network. An attacker can thus not trace the intended recipient of a message. Therefore an unconditionally reliable broadcasting of a message in a DC-network ensures an **unconditional recipient untraceability**.

Unconditional recipient untraceability

If the sender of a message in DC-network will keep it secret from all participants but the intended recipient, then he encrypts it with the intended recipient's public key. Thereafter, he appends it with random bits and broadcasts it over all participants. An observer of the message traffic can once again not find out the intended recipient. The intended recipient receives the message decrypts it with his private key and can see it. He is the only one able to do this

Key establishment

Pseudorandom-sequence generator

As mentioned before it is suggested that each any of two participants of a DC-network must share a key. The question now is how can these participants establish the necessary keys? A **key establishment** – based on common key establishment protocols – would be well costly, since a large portion of keys, having the same length as the messages, must be permanently exchanged. Alternatively, each two participants can share a short key and then utilize a cryptographic **pseudorandom-sequence generator** to expand it for each use. In this case, the protocol would not be secure in an information-theoretical sense any more but rather in a complexity-theoretical sense. Another possibility to establish keys between participants is that each two of them agree an a common file of random bits (key bits) and store them on two optical disks[34]. Each one is also associated to one participants who has therewith enough keys for sending messages.

Exercise 5.3-1:

*Explain the notions **Sender anonymity**, **Server anonymity**, **Recipient anonymity** and **Client anonymity**.*

34 On CD ROM for example.

6 Packet filters and firewall systems

In Chapter 5 we have discussed the disadvantages and advantages of host and network security. With host security you enforce the security of each host machine separately and make the effort to avoid or alleviate all the known security problems that might affect that particular host. Host security is hard to achieve and does not scale in the sense that as the number of hosts increases, the ability to ensure that security is at a high level for each host decreases. On the other side, a network security model concentrates on controlling network access to your various hosts and the services they offer, rather than on securing them one by one. Network security approaches include building intermediate systems to protect your internal systems and networks, using strong authentification approaches, like public-key or one-time passwords, and using encryption and integrity checks to protect particularly sensitive data as it transits the network through routers.

In this chapter, we discuss possible solutions to get network security with packet filter and/or firewalls (firewall systems). There are many different definitions of the term ”firewall” in the literature. A firewall represents a blockade between a privately owned and protected intranet, that is assumed to be secure and its users are trusted, and another network, typically a publicly owned network or the Internet (see Fig. 6-1). The later is assumed to be nonsecure and untrusted. The purpose of the firewall is to prevent unwanted and unauthorized communications into or out of the protected network.

In [CB94] a firewall system is defined as a collection of components placed between two networks that collectively have the following properties:

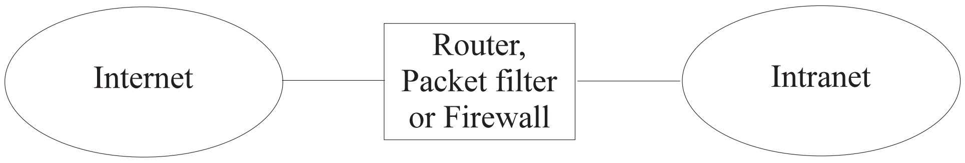


Fig. 6-1: Simplified topology of a packet filter or firewall system.

1. All traffic from inside to outside, and vice versa, must pass through the firewall.
2. Only authorized traffic, as defined by the local security policy, will be allowed to pass.
3. The firewall system itself is immune to penetration.

These properties are design goals. A failure in one aspect does not necessarily means that the collection is not a firewall, but simply that it is not a good one. Consequently, there are different grades of security that a firewall can achieve. Note that there must be a local security policy when the rules of the firewall systems are established. Other definitions of firewall systems include connections and data streams from inside to outside and vice versa and must be strongly authenticated by users. In this case, the firewall system has to operate at higher layers in a communication protocol stack where information about users is available. These systems require the use of application gateways. Systems which only operate at lower layers without authentification information about the end users or connections are called packet filters. For the sake of clarity, in this chapter we make a clear distinction between packet filters (operating at the network layer, Internet layer or transport layer in the Internet protocol stack) and firewalls operating at some higher layer. The definitions can also be applied to none Internet based protocol stacks. The firewall implements parts of a network security policy by forcing all traffic from inside to outside, and vice versa, to be directed or routed to the firewall, where it can be examined and evaluated accordingly. A firewall also requires its users to strongly authenticate themselves before any further action is deployed.

Often there is also made a difference between the terms ”firewall technology” and ”firewall system” ([Sch97]): A firewall technology refers to a set of mechanisms that can enforce a network security policy on communication traffic entering or leaving a domain, whereas a firewall system is an instantiation of the firewall technology.

inbound, outbound

In the literature the terms **inbound** and **outbound** are used in conjunction with the words connection, packet and interface. We will define the use of these in the following way: An outbound connection is a connection initiated from a client on an internal machine to a server running on an external network. An outbound connection includes outbound packets, travelling from the client to the external server, and inbound packets from the server to the internal client. The inbound interface for a packet refers to the physical network interface on a router that the packet actually appeared, while the outbound interface refers to the physical network interface the packet will go out on if it isn’t denied by the application of a specific packet filter rule.

Firewall systems also have disadvantages. Probably the main disadvantage is that a firewall can not protect an intranet against inside attacks. For that matter, internal firewalls may be used to control access between different administration and security domains, or to protect sensitive parts of a corporate intranet. Internal firewalls are sometimes called intranet firewalls. From a purely technical point of view, there is nothing that distinguishes an intranet firewall from an Internet firewall except for the policy it enforces. Consequently, we are not going to differentiate between intranet and Internet firewalls in the rest of this chapter.

Packet filters and stateful inspection of connections are addressed in Section 6.1. Packet filters work usually at the Internet layer and transport layer in the Internet protocol model (see Fig. 6.1-2). In Section 6.2 circuit-level gateways are described, which are fundamentally different from packet filtering. Application-level gateways and proxy servers are discussed in Section 6.3 and possibilities for network topologies and architectures in Section 6.4.

6.1 Packet filtering

router

A **router** is an internetworking device that runs an operating system to transfer packets between two or more physically separated network segments. It operates at the network layer in the OSI reference model, or the Internet layer in the Internet protocol model. Most routers today are able to route multiple protocols, like IP packets, IPX packets, etc, but we focus on IP routing. A router transports IP packets by consulting routing tables that indicate the best path the packet should take to reach its final destination. More precisely, a router receives an IP packet on one network interface (inbound) and forwards it to another network interface (outbound). If the router is not able to transport the packet, maybe because of an invalid destination IP address or an errorneous IP header, it sends an **Internet Control Message Protokoll** (ICMP) message to the source IP address.

Internet Control Message Protokoll, ICMP

packet filter

A **packet filter** is an internetworking (multihomed) device that applies a set of rules to each incoming or on that device generated packet in order to decide whether and how it will be forwarded or dropped. Specially IP packets are filtered based on information usually found in packet headers, such as

1. inbound network interface or locally generated packet,
2. source and destination IP address,
3. the transported service (UDP, TCP, ICMP),
4. values of IP, TCP or UDP flags and options.

Routers that support packet filtering are sometimes referred to as screening routers. Many commercial router products provide the capability to screen IP packets and to filter them in accordance with a set of packet filter rules.

stateless packet filter

Packet filters work **stateless**. This means, that any packet is examined isolated from what happened in the past, forcing the packet filter to make decisions to permit or deny each packet based upon the packet filter rules. This behaviour is sometimes also referred to as static packet filtering, meaning that the packet filter rules are static and context insensitive.

6.1.1 Packet filters and the IP protocol family

There exist some special problems that have do be considered when you want to protect your internal IP network with a packet filter or a firewall ([ZCC00]):

stateless source routing

1. **Source Routing option in the IP header**

   The most common IP option a firewall would be confronted with is the IP source route option. Source routing lets the source specify the route the packet is supposed to take to its destination, rather than letting each router along the way use its routing table to decide where to send the packet next. In practice, source routing is commonly used only by attackers attempting to circumvent security measures by causing packets to follow unexpected paths. This is in fact a circular problem. Some packet filtering systems take the approach of dropping any packet that has any IP option set.

packet fragmentation

2. **Fragmentation of IP packets**

   From a packet filtering point of view, the problem with fragmentation is that only the first fragment will contain the header information from higher layer protocols, like TCP, that the packet filtering system needs in order to decide whether or not to allow the full packet to be transmitted. Originally, the common packet filtering approach to deal with fragmentation was to allow any non-first fragments through and to do only packet filtering in the first fragment of a packet. This was considered safe because if the packet filtering decides to drop the first fragment, the destination system will not be able to reassemble the rest of the fragments into the original packet, regardless of how many of the rest of the fragments it receives. If it can not reconstruct the original packet, the partial packet will not be accepted.

   However, there still remain problems with fragmented packets. If you pass all non-first fragments, the destination host will hold the fragments in memory for a while, waiting to see if it gets the missing piece. This makes it possible for attackers to use fragmented packets in a denial of service attack. When the destination host gives up on reassembling the packet, it will send an ICMP message (”of type ”packet reassembly time expired”) back to the source host, which will tell an attacker that the host exists and why the connection did not succeed.

   In addition, attackers can use specially fragmented packets to conceal data. Each fragment contains information about where the data it contains starts and ends. Normally, each one starts after the last one ended. However, an attacker can construct packets where fragments actually overlap, and contain the same data addresses. This does not happen in normal operation. It can only happen when bugs or attackers are involved.

   Because of these many and varied problems with fragmentation, you should look for a packet filter that does reassembly. Rather than either permitting or denying fragments, the packet filter should reassemble the packet locally, and if necessary, refragment it before sending it further. This will increase the load on the packet filter and/or firewall somewhat, but it protects against all fragmentation-based risks and attacks, except those the packet filter itself is vulnerable to.

If the packet filter can not do fragment reassembly, the safest option is to reject all non-first fragments. This may destroy connections that otherwise would have succeeded, but it is the lesser of two evils. Denying fragments will cause some connections to fail mysteriously, which is extremely unpleasant to debug. On the other hand, allowing them will open a variety of attacks that are widely exploited on the Internet. Fortunately, fragmented packets are becoming rarer as the use of path MTU discovery increases.

3. **TCP connections**

TCP connections

TCP is connection oriented and bidirectional in that once a connection is established, a server can reply to a client over the same connection. If TCP connections should be blocked, it is sufficient to simply block the first packet (setup) of the connection. Without that first packet, any further packets in that connection can not be reassembled into a data stream by the receiver, and the connection will never be made. That first packet is recognizable because the ACK bit in its TCP header is not set and the SYN bit is set. All other packets in the connection, regardless of which direction they are going in, will have the ACK bit set.

Recognizing these ”start-of-connection” TCP packets lets you enforce a policy that allows internal clients to external servers but would not allow external clients to internal servers. Attackers cannot subvert this approach simply by turning on the ACK bit in their start-of-connection packets, because the absence of the ACK bit is what identifies these packets as start-of-connection packets.

Packet filtering implementations vary in how they treat and let you handle the ACK bit. Some packet filtering implementations give direct access to the ACK bit. Some other implementations give indirect access to the ACK bit. Finally, some implementations do not let you examine the ACK bit at all.

6.1.2 Packet filtering for FTP sessions

There are some practical problems related to the use of static packet filtering, for example with the FTP protocol. Consider an FTP session initiated by a client running on an internal machine, which is inside the intranet. The FTP server resides on the Internet. The FTP session basically consists of two TCP connections:

1. An outbound control connection initiated by the client.

2. An inbound data connection initiated by the server.

The outbound connection is initiated by the client from a random TCP port to the FTP server control port 21. The inbound connection for data transfer is initiated by the FTP server data port 20 to a specific port on the client side. More precisely, the client informs the server about the destination port number it should use to connect to for the inbound connection by using an FTP PORT command. To open the data connection, the server establishes a TCP connection to that particular port. Consequently, the packet filtering device will see an inbound TCP segment trying to establish an inbound connection. Most packet filters are configured to discard these kinds of TCP segments and drop the corresponding IP packets accordingly.

From the packet filter's point of view, the problem of FTP exists because the data connection is inbound, meaning that it is established from an external server to an internal client. The problem is trivially solvable if both connections (the FTP control connection and the FTP data connection) are established outbound, meaning that they are initiated by the client.

passive mode FTP

Following this line of argumentation, **passive mode FTP** has been proposed as a preliminary solution in RFC 1579 ([Bel94]). In passive mode FTP, instead of sending out an FTP PORT command, the client waits for the server to specify a particular port number and establishes an outbound TCP connection to that port. As a result, any packet filter sitting between the client and the server experiences two outbound TCP connections (instead of one outbound and one inbound connection, as in the case of normal mode FTP). In general, the network security policy that drives the packet filtering device allows outbound connections to be established and tolerates passive mode FTP accordingly.

In summary, passive mode FTP represents a viable solution for intelligently filtering FTP traffic. It is however, not a general solution and other application protocols may come up with different problems that can not be addressed with passive mode.

6.1.3 Stateful inspection

A more general solution to filter connection oriented services is called stateful inspection. It was suggested to introduce some state into the packet filtering process. Consequently, the resulting packet filter rules are dynamic and context-sensitive, the first attribute also being reflected in the term **dynamic packet filtering** that is sometimes used as a synonym for stateful inspection.

dynamic packet filtering

stateful inspection

Stateful inspection (or dynamic packet filtering) looks at the same header information as static packet filtering does, but can also peek into the payload data where the transport and application layer data usually appears. More importantly, stateful inspection maintains state information about passed IP packets. It compares the first packet in a TCP connection to the packet filter rules, and if the packet is permitted, state information is added to an internal database. Think of this state information representing an internal virtual circuit in the firewall on top of the transport layer association. This information permits subsequent packets in that association to pass quickly through the firewall. If the rules for a specific type of service require examining application data, the part of each packet must still be examined.

For example, a dynamic packet filtering device can react on seeing an FTP PORT command by creating a dynamic rule permitting a TCP connection back from the FTP server to that particular port number of the client side. Logging, or authentication as required by the rules, generally occurs at the application layer. Although the opportunity for better logging is present, stateful inspection firewalls typically only log the source and destination IP addresses and port numbers, similar to logging with a packet filter or screening router.

In spite of the fact that you can introduce state information to improve the capabilities of a packet filtering device, the problem remains that there is no such thing as an association between a data stream and a previously authenticated and authorized user. To make things worse, there is no such thing as a user on the Internet layer where packet filtering occurs. Consequently, true firewalls must operate above the Internet layer, typically at the transport layer.

There are also further problems with stateful packet filtering:

1. State tracking of a packet filter provides the ability to do things that you cannot do therewise, but it also adds complications. First, the packet filter has to keep track of the state. This increases the load on the packet filter, opens it to a number of denial of service attacks, and means that if the router reboots, packets may be denied when they should have been accepted. If a packet may go through redundant packet filters, they all need to have the same state information. There are protocols for exchanging this information, but it is still a tricky business. If you have redundant routers simultaneously, the state information needs to be transferred between them almost continously, or the response packet may come through before the state is updated.

2. Second, the packet filter has to keep track of state without any guarantee that there is ever going to be a response packet. Not all UDP packets have responses. At some point, the packet filter needs to give up and to get rid of the rule that will allow the response. If the packet filter gives up too early, it will deny packets that should have been accepted, causing delays and unneeded network traffic. If the router keeps the rule too long, the load on the router will be unnecessarily high, and there is an increased chance that packets will be accepted when they should have been denied. Some protocol specifications provide guidelines, but these are not necessarily useful. For instance, DNS (Domain Name Service) replies are supposed to arrive within 5 seconds, but reply times for name service queries across the Internet can be as high as 15 seconds. An implementation of the protocol specification will almost surely deny a response that you wanted to be accepted.

6.1.4 Network Address Translation (NAT)

Network Address Translation, NAT

Network Address Translation (NAT) allows a network (intranet) to use one set of network addresses internally and a different set when dealing with external networks. NAT does not, by itself, provide security, but it helps to conceal the internal network layout and to force connections (from inside to outside, and vice versa) to go through a choke point. NAT, routing, and packet filtering can be done by one host, between the intranet and the Internet. Only the NAT host has an official and valid Internet address. The addresses of the other hosts of the intranet (private network) have IP addresses that are not public to the Internet. An address allocation scheme for intranets is described in RFC 1918 ([RM+96]).

NAT is standardized in RFC 2766 [TS00] by the IETF. For further information we also refer to [SH99] and [SE01].

Port and Address Translation, PAT

Like packet filtering, NAT works by having a router to do extra work. In this case, the router not only sends packets, but also modifies them. When an internal host sends a packet to the outside, the network address translation system modifies the IP source address of the IP packet to make the packet look as if it is coming from a valid Internet address. When an external host sends a packet to the inside, the network address translation system modifies the destination address into the correct internal address. The network address translation system can also modify the source and destination port numbers. This is also called as **Port and Address Translation** (PAT).

Network address translation systems can use different schemes for the translation between internal and external addresses:

1. Allocation of one external host address for each internal host and always apply the same translation: This technique provides no savings in address space, and it slows down connections. It is normally a temporary measure used by sites that have been using illegal address space but are in the process of moving to using valid addresses.

2. Dynamic allocation of an external host address each time an internal host initiates a connection, without modifying port numbers: This limits the number of internal hosts that can simultaneously access the Internet to the number of available external addresses.

3. Creation of a fixed mapping from internal IP addresses to externally visible addresses, but with the use of port mapping so that multiple internal hosts use the same external IP addresses but with a different port number.

4. Dynamically allocation of an external host address and port pair each time an internal host initiates a connection. This makes the most efficient possible use of the external host addresses.

There exist also some security advantages in the use of NAT:

1. NAT helps to enforce the packet filter and firewall control over outbound connections.

2. NAT can help to restrict incoming traffic, if dynamic allocation schemes are used.

3. NAT helps to conceal the internal network configuration.

While NAT is a very useful way of conserving network address space, it presents some problems:

1. Dynamic allocation requires state information that is not always available for the NAT system.
2. Embedded IP addresses in the packet payload are a problem for NAT.
3. NAT interferes with some encryption and authentication protocols including IPsec (see Chapter 3), which protects the entire IP packet including the IP header.
4. Dynamic allocation of addresses interferes with logging.
5. Dynamic allocation of ports may interfere with packet filtering.

The NAT protocol is an extension module in the packet filter program iptables (see Section 6.1.5).

masquerading

We will now discuss **masquerading** which is a form or system of network address translation. Because it is capable of working at higher protocol levels, and doing more intricate modifications than simple address changes, it is also called a **transparent proxying system**. What it does could be considered either proxying or packet filtering: it is somewhere in between the two.

transparent proxying system

The IP address of the packet filter or firewall system is used to communicate with remote services. For simple protocols, masquerading alters only IP header information, including IP addresses, port numbers, and TCP sequence and acknowledgement numbers. Masquerading uses the IP address of the host (packet filter or firewall) doing the masquerading as the externally visible address, and maps the port number into one from a fixed pool of ports. Masquerading works by intercepting packets that are being forwarded by the packet filter. Masquerading for simple protocols works much like simple network address translation. IP addresses and port numbers are modified on outgoing packets. For TCP connections, a new sequence number is generated. The process is reversed for incoming packets.

Example 6.1-1: Masquerading of an internal service
Suppose we have an internal network `141.161.1.0` with a subnet mask `255.255.255.0`. In this internal network there is a host (IP number `141.161.1.3`) with a Telnet server `S` running on TCP port `22`. The internal network is protected with a packet filter with routing and masquerading functionality. This packet filter `P` has two network interfaces, one connected with the internal network and one connected with the Internet (IP number `172.16.3.5`). The internal network is not visible from the Internet. The masquerading system is so configured, that incoming TCP packets with destination IP number `172.16.3.5` and destination port `22` are modified and forwarded in the internal network with the destination IP address `141.161.1.3` and the destination port number `23`. If a Telnet client `C` on the Internet (for example with IP number `123.67.5.4`) sends a TCP connection request to `172.16.3.5`

and destination port `23`, this request is forwarded to `141.161.1.3` and port `23`.

The first two steps of the connection (request and response) are as follows, where `X` is an arbitrary port number ≥ 1024:

1. $C \rightarrow P$:
   (source IP=`132.67.5.4`, source port=`X`, dest. IP=`172.16.3.5`, dest. port=`23`)

2. $P \rightarrow S$:
   (source IP=`132.67.5.4`, source port=`X`, dest. IP=`141.161.1.3`, dest. port=`23`)

3. $S \rightarrow P$:
   (source IP=`141.161.1.3`, source port=`23`, dest. IP=`132.67.5.4`, dest. port=`X`)

4. $P \rightarrow C$:
   (source IP=`172.16.3.5`, source port=`23`, dest. IP=`132.67.5.4`, dest. port=`X`)

6.1.5 Packet filters for the Linux operating system

In this section we describe how to configure the most well known, free available, and recent packet filter for the Linux operating system, namely `iptables`.

ipfwadm
ipchains
iptables

If you want to build a firewall system for the Internet protocols on a Linux platform, it depends on the version of the kernel of your Linux system. If the kernel version is prior to 2.2, you should use the **ipfwadm** utility. Kernels with version 2.2.x support the third generation program of packet filters, called **ipchains**. Linux kernels 2.3.15 and later support the fourth generation of Linux packet filters called **iptables**. All three tools can be used to build a packet filter by defining rules which are applied to IP packets. The Linux kernel must be configured to support IP packet filtering. This could be done by selecting the appropriate options when performing the `make menuconfig` command in the process of compiling the Linux kernel.

The use and configuration of the tools `ipfwadm` and `ipchains` are described in [KD00] (chapter 9) and in [ZCC00]. We will focus on the newest developments of IP packet filters for the Linux operating system: `iptables`.

netfilter

While developing `ipchains`, Paul Russel decided that IP packet filtering should be less difficult on the Linux operating system. He soon set about the task of simplifying aspects of packet processing in the Linux kernel packet filtering code and produced a filtering framework that was transparent and much more flexible. He called this new framework **netfilter**. This concept was implemented and a new and extensible configuration and management tool `iptables` was created[35].

`Iptables` is used to set up, maintain and inspect the tables of IP packet filter rules in the Linux kernel. There are several different tables (`filter`, `nat`, `mangle` and user-defined), and each table contains a number of built-in chains, and may contain user-defined chains. Each chain is a list of rules which can match a set of packets: each rule specifies what to do with a packet which matches. This is called the **target** of a packet, which may be an action like accepting, dropping, modifying a packet or to jump to a user-defined chain in the same table or a return to the calling chain:

targets

1. `ACCEPT` means to let the packet through.
2. `DROP` means to drop the packet.
3. `QUEUE` means that the packet is passed to a user-defined chain.
4. `RETURN` means stop traversing this chain, and resume at the next rule in the previous calling chain.

If the packet does not match a rule, the next rule in the chain is examined. If the end of a user-defined chain is reached, or a rule in a built-in chain with target `RETURN` is matched, the target specified by the chain policy determines the fate of the packet.

Fig. 6.1-1 shows the different ways of packets through the system of a host and the points P_i, $1 \leq i \leq 5$, where iptables can interact with the packets. The packets can be generated by a local process or are arriving at a network interface of the host (at the left side of Fig. 6.1-1). The packets are leaving the host system through a network interface (at the right side of Fig. 6.1-1) or end at a local process, if they are not dropped by a rule of `iptables`.

There are currently three independent tables, which are present depending on the kernel configuration options and the dynamically loaded modules. The **tables** and chains are as follows (see Fig. 6.1-2):

tables and chains

1. **filter**: This is the default table, and contains the built-in chains `INPUT` at P_4 for packets to a local process, `FORWARD` at P_2 for packets being through a network interface, and `OUTPUT` at P_5 for locally generated packets.
2. **nat**: This table can be used to do network address translation. It consists of three built-in chains: `PREROUTING` at P_1 for altering packets as soon as they come in the host, `OUTPUT` at P_5 for altering locally-generated packets before routing, and `POSTROUTING` at P_3 for altering packets as they are about to go out.

35 The official homepage of `netfilter` is http://www.netfilter.org/. There you can find code sources, documentation, FAQs, tutorials and presentations concerning `netfilter` and `iptables`

3. **mangle**: This packet is used for specialized packet alteration. It has two built-in chains: `PREROUTING` at P_1 for altering incoming packets before routing and `OUTPUT` at P_5 for altering locally-generated packets before routing.

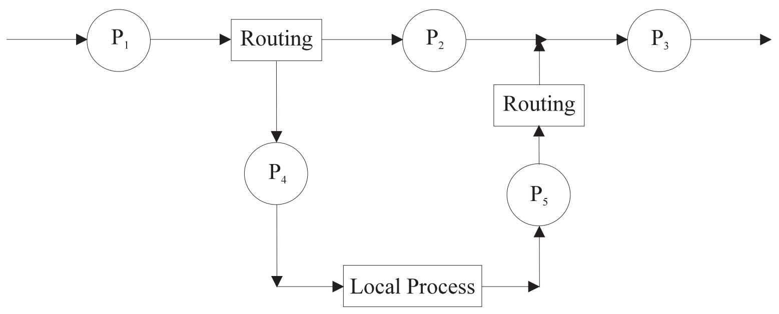


Fig. 6.1-1: This figure shows the different ways of a packet through the system of a host and the pointsP_i, $1 \leq i \leq 5$, where `iptables` can interact with the packets.

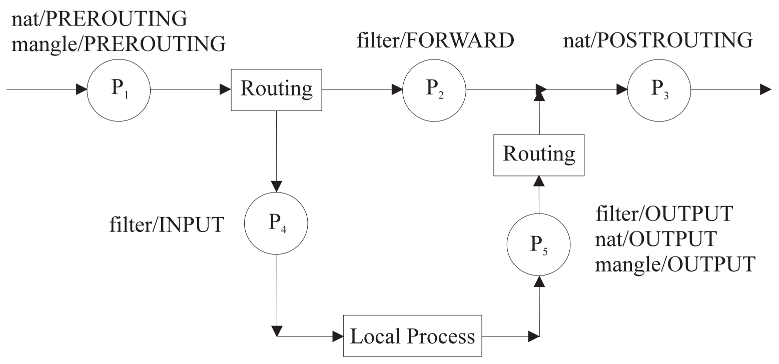


Fig. 6.1-2: Tables and chains of `iptables`.

With the command `iptables` the chains can be defined, extended and managed. A call has the following structure:

```
iptables -t table command chain parameters options
```

where `table` is from **filter** (default), **nat** or **mangle**, and `command` is a command to modify the `chain` in the selected table. For the long versions of the command, parameter and option names one only need to use enough letters to ensure that `iptables` can differentiate it from all other options.

The following commands exist in iptables:

- **-A**, **--append**: Append one or more rules to the end of the selected chain. The rules in a chain are numbered upwards from one.
- **-C**, **--check**: Check the datagram described by the rule specification against the specific chain. This command will return a message describing how the chain processed the datagram.
- **-D**, **--delete**: Delete one or more rules from the selected chain.
- **-R**, **--replace**: Replace a rule in the selected chain.

- **-I**, **--insert**: Insert one or more rules in the selected chain as the given rule number. The rule number has to be specified after the parameter chain.
- **-L**, **--list**: List all rules in the selected chain. With the option **-v** or **--verbose** a verbose output is produced.
- **-F**, **--flush**: Flush the selected chain. This is equivalent to delete all rules.
- **-Z**, **--zero**: Zero the packet and byte counters in all chains.
- **-N**, **--new-chain**: Create a new user-defined chain of the given name.
- **-X**, **--delete-chain**: Delete the specified user-defined chain of the given name.
- **-P**, **--policy**: Set the policy for the given chain. Valid policies are `ACCEPT`, `DROP`, `QUEUE`, `RETURN`. `ACCEPT` allows the datagram to pass. `DROP` causes the datagram to be discarded. `QUEUE` causes the datagram to be passed to a user-defined chain for further processing. `RETURN` causes the packet filter code to return to the chain that called the one containing this rule, and continue starting at the rule after the calling rule.
- **-E**, **--rename-chain**: Rename the user specified chain.
- **-h**, **--help**: Help on the command line syntax of `iptables` or if specified, to an extension module.

The following parameters exist. The symbol ”!” can be understood as a negation.

- **-p**, **--protocol**`[!]prot`
  The protocol of the rule. The variable prot can be one of `tcp`, `udp`, `icmp`, `all` (default) or a numeric value, representing a protocol over IP.
- **-s**, **--source**`[!]address[/mask]`
  Specification of the source IP number. The value of `address` can be a hostname, a network name, or a plain IP address. The `mask` can be either a network mask or a plain number, specifying the number of 1’s at the leftside of the network mask. Thus, a mask of `24` is equivalent to `255.255.255.0`
- **-d**, **--destination**`[!]address[/mask]`
  Specification of the destination IP number.
- **-j**, **--jump**`target`
  This specifies the `target` of the rule, i.e. what to do if the packet matches it. The `target` can be a user-defined chain (not the one this rule is in), one of the special built-in targets which decide the fate of the packet immediately, or an extension. If this option is omitted in a rule, then matching the rule will have no effect on the packet’s fate, but the counters on the rule will be incremented.
- **-i**, **--in-interface**`[!]name`
  `Name` of an interface via which a packet is received. Only appliable for the

INPUT, FORWARD and PREROUTING chains. If the interface name ends with a ”+”, then any interface which begins with this name will match.

- **-o**, **--out-interface**[!]name
  Name of an interface via which a packet is going to be sent. Only appliable for the FORWARD, OUTPUT and POSTROUTING chains. If the interface name ends with a ”+”, then any interface which begins with this name will match.

- [!]**-f**, **--fragment**
  This means that the rule only refers to second and further fragments of fragmented IP packets. When the ”!” argument preceeds the ”**-f**” flag, the rule will only match head fragments, or unfragmented packets.

Iptables can use extended packet matching modules. These modules can be loaded in two ways, when **-p** or **--protocol** is specified, or explicitly with the **-m** or **--match** options, followed by the matching module name. After these, various extra command line options become available, depending on the specific module. Besides other, the following module names and protocols are possible:

1. tcp for TCP packets,
2. udp for UDP packets,
3. icmp for ICMP packets and
4. max for specifications for packets on the network layer.

For TCP packets (**--protocol**tcp), the following options are possible:

- **--source-port**[!][port[:port]]
  TCP source port specification. This can be a service name or a port number. An inclusion range can also be specified, using the format port:port. If the first port is omitted, ”0” is assumed. The flag **--sport** is an alias for this option.

- **--destination-port**[!][port[:port]]
  TCP destination port specification. The flag **--dport** is an alias for this option.

- **--tcp-flags**[!]mask comp
  Match when the TCP flags are as specified. The first argument (mask) is the flags which should examine, written as comma-separated list, and the second argument (comp) is a comma-separated list of flags which must be set. Flags are: SYN, ACK, FIN, RST, URG, PSH, ALL, NONE.

- [!]**--syn**
  Only match TCP packets with the SYN bit set and the ACK and FIN bits cleared. Such packets are used to request TCP connection initiation. For example, blocking such packets coming in an interface will prevent incoming TCP connections, but outgoing TCP connections will be unaffected. This option is equivalent to **--tcp-flags**SYN,RST,ACK,SYN. If the ”!” preceeds the ”**--syn**”, the sense of the option is inverted.

- **--tcp-option**`[!]number`
  Match the packet, if the TCP option `number` is set.

For UDP packets (**--protocol**`udp`), the following options are possible:

- **--source-port**`[!][port[:port]]`
  UDP source port specification. The flag **--sport** is an alias for this option.

- **--destination-port**`[!][port[:port]]`
  UDP destination port specification. The flag **--dport** is an alias for this option.

For ICMP packets (**--protocol**`icmp`), the following option is possible:

- **--icmp-type**`[!]typename`
  This allows the specification of the `ICMP type`, which can be a numeric ICMP type, or one of the ICMP type names shown by the command `iptables -p icmp -h`.

The following option is possible for protocols in the network layer:

- **--mac-source**`[!]address`
  MAC address specification. It must be of the form `XX:XX:XX:XX:XX:XX`. Note that this only makes sense for packets entering the `PREROUTING`, `FORWARD`, `INPUT` chains for packets coming from an ethernet device.

There exist a lot of extension modules for logging of packets and actions, multiport definitions, to attempt to match various characteristic of the packet creator, state tracking and network address translation

Example 6.1-2: Iptables rules for a small network
Now we want to discuss a small network scenario. Suppose our intranet has the subnet IP number `172.16.1.0` with subnet mask `255.255.255.0`, which is equal to `24` ones from the left side. The users in the subnet are allowed to access WWW servers on the Internet with their WWW browsers, but no other traffic is allowed. Remember, WWW servers listen usually on port `80` of the TCP protocol for connection requests.

A screening router with Linux operating system (kernel $\geq$ 2.4) and installed `iptables` is physically between the Internet and the local intranet. The screening router has two network interfaces, one is connected with the Internet and the other with the intranet.

The `netfilter` kernel module has to be configured to fulfill the above policy for the intranet.

At first the kernel module that provides support for `netfilter` has to be loaded in the kernel:

```
$ modprobe ip tables
```

Now the commands to build the chains for the `filter` table, that fulfill the above policy:

```
$ iptables -F OUTPUT
$ iptables -P OUTPUT DROP
$ iptables -F INPUT
$ iptables -P INPUT DROP
$ iptables -F FORWARD
$ iptables -P FORWRAD DROP
$ iptables -A FORWARD -m tcp -p tcp -s 0/0 --sport 80
-d 172.16.1.0/24 --syn -j DROP
$ iptables -A FORWARD -m tcp -p tcp -s 172.16.1.0/24
-d 0/0 --dport 80 -j ACCEPT
$ iptables -A FORWARD -m tcp -p tcp -s 0/0
--sport 80 -d 172.16.1.0/24 -j ACCEPT
```

The table `filter` has not to be specified, because `filter` is the default table. With the first six commands all three chains in the `filter` table are cleaned and the default policy of the chains is set to `DROP`.

Exercise 6.1-1:

1. *Is a packet filter in our definition a firewall system? Give reasons!*
2. *What is a security policy and what should it include?*
3. *How do we call routers that support packet filtering?*
4. *What problems arise with the fragmentation process of IP packets, if you want to protect your IP network with a packet filter?*

6.2 Circuit-level gateways

In general networking terminology, an application gateway or gateway refers to an internetworking device that connects one network to another for a specific application. Therefore, the gateway must understand and implement the corresponding application protocol. In the client-server model, a gateway refers to an intermediate process running between the client that requests a particular service and the server that provides the service. In this model, the gateway functions as a server from the client's point of view, and as a client from the server's point of view. In general, a gateway either works at the application layer or at the transport layer ([Opp00]):

1. If a gateway works at the application layer, it is usually called an application-level gateway, or proxy server in short (see Section 6.3).

2. If a gateway works at the transport layer, it is usually called a circuit-level gateway.

Most gateways used in firewall configurations work at the application layer and actually represent application-level gateways or proxy servers. In either case, the application gateway runs on a firewall host and performs a specific function as a proxy on the user's behalf.

The idea of a circuit-level gateway is fundamentally different from packer filtering (either static or dynamic). In essence, a circuit-level gateway takes a TCP connection request form a client, authenticates and authorizes the client, and establishes a second TCP connection to the origin server on the client's behalf. After having successfully established this second TCP connection, the circuit-level gateway simply relays data forth and back between the two connections. In particular, it does not interfere with the data stream. This differentiates a circuit-level gateway from an application-level gateway or proxy server that is able to actually understand the application protocol employed by the two endpoints of the connection.

SOCKS

The single circuit-level gateway that is actually in widespread use is **SOCKS**. SOCKS[36] follows a customized client approach, meaning that it requires customizations and modifications to client software, but no change is usually required to user procedures. More precisely, SOCKS requires modifications either to the client software or to the TCP/IP stack to accommodates the interception at the firewall between the client and the server:

1. A client that has been modified to handle SOCKS interactions is commonly referred to a socksified client. Following this terminology, Netscape Navigator is a socksified HTTP client, as it accommodates interactions with SOCKS server. A socksified client issues SOCKS calls that are transparent to the users.

2. Socksified TCP/IP stacks are also available, which may obviate the need for client software modifications.

In either case, the SOCKS server resides at the firewall host and interacts with the socksified clients or TCP/IP stacks. There are no further changes required for the servers.

SOCKS and the SOCKS protocol for communications between a socksified client and a SOCKS server were originally developed by David and Michelle Koblas ([KK92]). There are several software packages publicy and freely available on the Internet. A package consist of two components:

1. A SOCKS server, called `socksd`, and

2. a SOCKS library that can be used to replace regular TCP/IP function calls in the client software.

36 The homepage of SOCKS is http://www.socks.permeo.com/

The design goal of SOCKS was to provide a general framework for TCP/IP applications to securely use the services of a firewall. Complying with these design goals, SOCKS is independent of any supported TCP/IP application protocol. When a socksified client requires access to an Internet server, it must first open a TCP connection to the appropriate port on the SOCKS server residing on the firewall system. The SOCKS server is conventionally located on TCP port 1080. If the TCP connection is established, the client sends a connection relay request to the SOCKS server. The request includes the following information: desired (IP) destination address, desired destination port and authentication information.

The SOCKS server evaluates the information from the client in the connection relay request. During this evaluation, it may perform various functions, such as authentication, authorization, message security-level negotiation, and so on. The SOCKS server either accepts the request and establishes a corresponding connection to the Internet application server, or rejects the request. The evaluation depends on the configuration data of the SOCKS server. Once the requested connection is establis-hed, the SOCKS server simply relays data between the client and the server.

There are currently two versions of SOCKS available: SOCKS version 4 and SOCKS version 5. SOCKS V4 has been widely used in firewall systems, but it does not use real user authentication. It bases its decisions on whether to allow or deny connections on the same sort of information that packet filters use. SOCKS V5 provides support for several different ways of authenticating users, which gives you more precise control and logging. The SOCKS V5 protocol is standardized by the IETF in RFC 1928 ([LGL+96]). Two authentification methods are specified:

1. Password-based authentification in [Lee96] and

2. Kerberos V5 GSS-API authentification in [McM96].

Further differences of Version 4 and 5 are described in [ZCC00] and [Opp00].

Finally, note that the only difference between a circuit-level gateway and a simple port forwarding mechanism is that with circuit-level, the client is aware of the intermediate system, whereas in the case of a port forwarding mechanism, the client may be completely oblivious of the existence of the intermediate. Also, a circuit-level is generic, and any TCP connection can be handled by the same gateway. Contrary to that, a port forwarding mechanism is usually specific to a specific service, meaning that all qualifying TCP segments are forwarded to a specific port of the destination server.

6.3 Application-level gateways and proxy servers

An application-level gateway or proxy-server is similar to a circuit-level gateway as explained in the previous section. The main difference is that the former understands the application protocol being relayed, whereas the second is generic and not application-specific. Consequently, a specific application-level gateway or proxy server is implemented for each application protocol a firewall must support. In the rest of this chapter, we use the terms application-level gateway and proxy server synonymously and interchangeably.

Typically, when a client contacts an application-level gateway using one of the TCP/IP application protocols, such as Telnet, FTP, or HTTP, the gateway asks for some valid user authentification and authorization information. In the easiest case, this information simply consists of a username and a corresponding password. However, if the proxy server is accessible from the Internet, it is strongly recommended to use some strong authentication mechanisms, such as provided by one-time passwords or challenge-response systems. If the proxy server accepts the user authentication and authorization information, it either connects to a preconfigured remote system to be accessed on the user's behalf, such as an FTP or WWW server. In the second case, it is up to the user to respond accordingly and to provide the name of a remote system. The proxy server, in turn, contacts the remote system and establishes a secondary TCP connection to that system. After having established the secondary TCP connection, the proxy server fully controls the data stream that is being relayed between the two connections. In particular, the proxy server can scan the data stream for specific protocol commands or data contents, as well as enforce specific restrictions on inbound and outbound traffic.

In order to properly authenticate a user, an application-level gateway or proxy server must have access to some identification and authentication information. In principle, this information can be either locally stored or remotely archived and made available through a security server.

The details on how proxying works differ from service to service. Some services provide proxying easily or automatically. For these services, you set up proxying by making configuration changes to normal servers. For most services, however, proxying requires appropriate proxy server software on the server side. On the client side it needs one of the following ([ZCC00]):

1. Proxy-aware application software
   With this approach, the software must know how to contact the proxy server instead of the real server when a user makes a request, and how to tell the proxy server what real servers to connect to.

2. Proxy-aware operating system software
   With this approach, the operating system that the client is running on is modified so that IP connections are checked to see if they should be sent to the proxy server. This mechanism usually depends on dynamic runtime linking. This mechanism does now always work and can fail in ways that are not obvious to users.

3. Proxy-aware user procedures
   With this approach, the user client software that does not understand proxying to talk to the proxy server tells the proxy server to connect to the real server, instead of telling the client software to talk to the real server directly.

4. Proxy-aware router
   With this approach, nothing on the client's end is modified, but a router intercepts the connection and redirects it to the proxy server or proxies the request. This requires an intelligent router in addition to the proxy server software.

TIS FWTK The **free firewall toolkit**[37] (**TIS FWTK**), from Trustd Information Systems, includes a number of proxy servers of various types. TIS FWTK also provides a number of other tools for authentication and other purposes. Whereas SOCKS attempts to provide a single, general gateway, TIS FWTK provides individual proxies for the most common Internet services. The idea is that by using small separate programs with a common configuration file, it can provide intelligent proxies that are safe, while still allowing central control. The result is an extremely flexible toolkit and a rather large configuration file. TIS FWTK includes:

- An authentication server that provides several mechanisms for supporting non-reusable passwords
- An access control program
- Proxy servers for a variety of protocols
- A generic proxy server for simple TCP-based protocols using one-to-one or many-to-one connections
- A wrapper for SMTP servers such as Sendmail to protect them form SMTP-based attacks
- A wrapper for inetd started servers to control where they can be contacted from.

6.4 Architectures and network topologies

In this section we discuss how to set up network topologies for packet filters or firewall systems to make an intranet secure.

In general, a firewall system consists of packet filters and application gateways (circuit-level or application-level). There are many possibilities to combine these components in firewall configurations. These are described in the next sections ([ZCC00], [Opp00]).

37 The homepage of TIS FWTK is http://freshmeat.net/projects/tisfirewalltoolkit/

6.4.1 Screening router

A simple topology (see Fig. 6.4-1) consists of a screening router, which connects the hosts on the intranet with the Internet. The screening router forwards or blocks packets, as determined by the site's security policy. This is a low-cost system, since you almost always need a router to connect to the Internet anyway, and you can simply configure packet filtering in that router. If the screening router is compromised or wrongly configured, you have no further security.

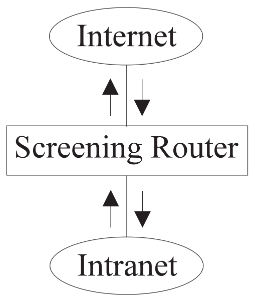


Fig. 6.4-1: Screening router.

A screening router is an appropriate firewall in situations where:

- The network being protected already has a high level of host security.
- The number of protocols being used is limited, and the protocols themselves are straightforward.
- You require maximum performance and redundancy.

6.4.2 Dual-homed host

A dual-homed host architecture (see Fig. 6.4-2) is built around a dualhomed host computer. A **dual-homed host** is a computer that has at least two network interfaces. Such a host could act as a router between the networks these interfaces are attached to. It is capable of routing IP packets from one network to another. However, to use a dual-homed host as a firewall, all routing functions are disabled. Thus, IP packets from one network are not directly routed to the other network. Systems inside (on the intranet) can communicate with the dual-homed host, and systems outside the firewall (on the Internet) can communicate with the dual-homed host, but these systems can not communicate directly with each other.

dual-homed host

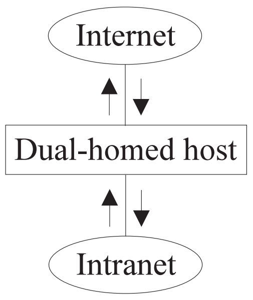


Fig. 6.4-2: Dual-homed host.

Dual-homed hosts can provide a very high level of control. If you aren't allowing packets to go between external and internal networks at all, you can be sure that any packet on the internal network has an external source is evidence of some kind of a security problem.

On the other hand, dual-homed hosts are not high performance devices. A dual-homed host has more work to do for each connection than a packet filter (screening router) does, and correspondigly needs more resources.

Since a dual-homed host is a single point of failure, it is important to make certain that its host security is absolutely impeccable. An attacker who can compromise the dual-homed host has full access to your site. An attacker who crashes the dual-homed host can cut you off from the Internet. Hosts with a high level of host security and firewall functions are called **bastion host**.

bastion host

The Internet community uses the term bastion host to refer to a bastion host computer system that is part of a firewall configuration and hosts for one or several application gateways. The term ”bastion” comes from the heavily fortified projections on the exteriors of castles in medieval times. A bastion host should be configured to be particularly secure, since it is exposed to direct attacks form the Internet (host security). Typically, a bastion host is located in a secure environment by residing on a secure operating system. In this case, the secure operating system must protect the firewall code and files from outside attacks.

6.4.3 Screened host

A better topology for a bastion host is shown in Fig. 6.4-3. In this architecture, the primary security is provided by packet filtering. The bastion host sits on the internal network. The screening router is configured to accept only traffic from the bastion host to the Internet and vice versa. The other hosts on the intranet can only communicate over the bastion host to the Internet. Any external system trying to access internal systems or services will have to connect to this host. The bastion host thus needs to maintain a high level of host security.

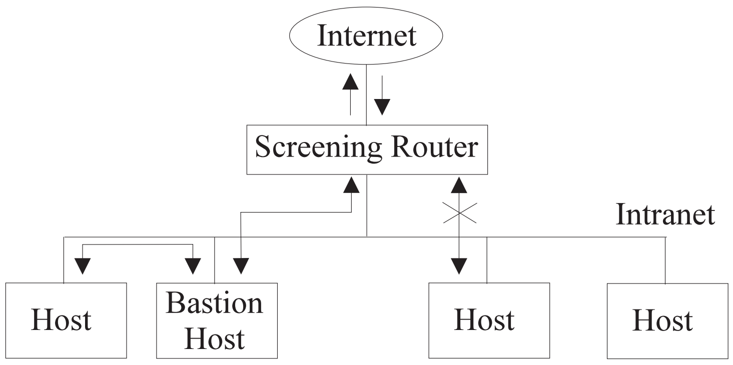


Fig. 6.4-3: Screened host.

For most purposes, the screened host architecture provides both better security and better usability than the dual-homed host architecture. Compared to other architectures, however, such as the screened subnet architecture (Section 6.4.4), there are some disadvantages of the screened host architecture. The major one is that if an attacker manages to break in to the bastion host, nothing is left in the way of network security between the bastion host and the rest of the internal hosts. The screening router also presents a single point of failure. If the screening router is compromised, the entire network is available to an attacker. For this reason, the screened subnet architecture, discussed next, has become increasingly popular.

A screened host architecture is appropriate when:

- Few connections are coming from the Internet. In particular, it is not an appropriate architecture if the screened host is a public WWW server.
- The network being protected has a relatively high level of host security.

6.4.4 Screened subnet

Fig. 6.4-4 illustrates the architecture of a screened subnet firewall[38]. In short, it consists of two screening routers used to create an outer and inner network segment. The outer network segment is also called a screened subnet or demilitarized zone (**DMZ**[39]). The DMZ may contain host application gateways running on one or more bastion hosts, as well as some additional servers that require carefully controlled Internet access, i.e. public WWW or FTP servers. Internal servers and hosts may be connected to the inner network segment.

DMZ

38 You can also find topologies in the literature, that are a little bit different from Section 6.4.4 and are named also as screened subnet.

39 The DMZ is named after the strip of no-man's-land between North and South Korea.

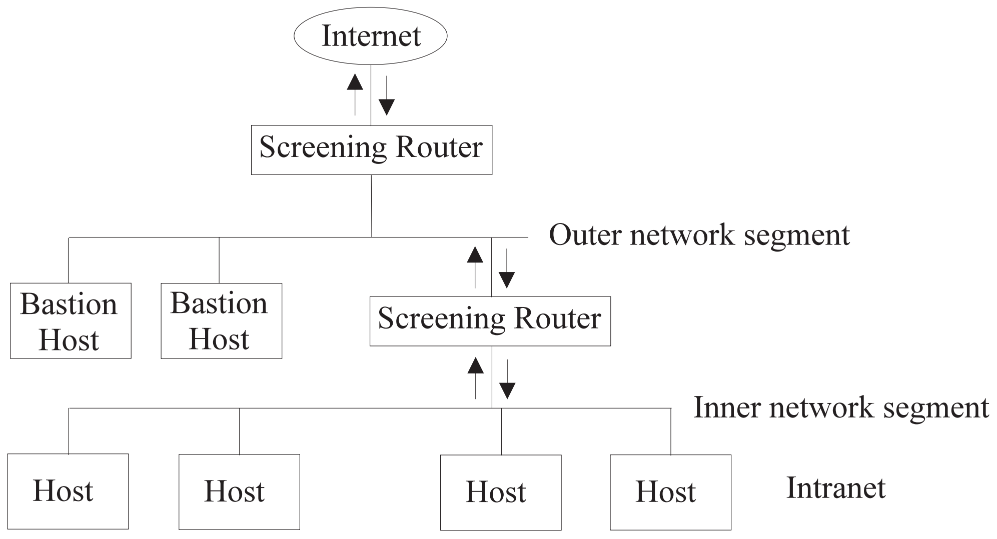


Fig. 6.4-4: Screened subnet.

In a split screened subnet topology the inner network segment is split by a screening router in two network segments.

In [ZCC00] and [Opp00] you can find more topologies and architectures for firewall systems.

6.5 Further Reading

We recommend the reading of the following books for the topic of firewalls: [KD00], [ZCC00], [Opp00] and [CB00].

There also exist many resources on the Internet:

- Internet Firewalls in general: Frequently Asked Questions maintained by Matt Curtin and Marcus J. Ranum:
  http://www.interhack.net/pubs/fwfaq/
- Netfilter homepage:
  http://www.netfilter.org
- SOCKS homepage and software download:
  http://www.socks.nec.com, ftp://ftp.nec.com/pub/socks

7 Application Layer Security

7.1 Introduction

Users interact at the application layer of the OSI reference model for communication. The application layer must be protected as vigorously as the other layers of the OSI model.

This chapter's focus is on security issues intrinsic to the application layer.

We will next overview all proposals concerning the security of Internet services and applications with the hope to find which layer is well suited to provide the security required (see Fig. 7.1-1):

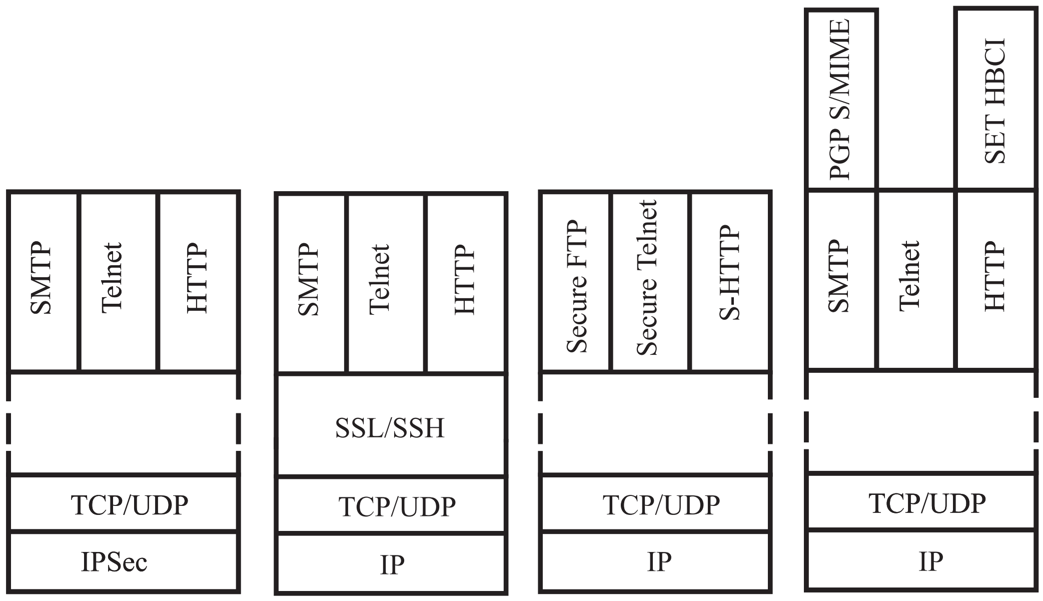


Fig. 7.1-1: Securing Internet services in different layers [FRU00].

1. Security protocols and mechanisms operating at the Internet layer:
   The **IP protocol** is the protocol operating at the Internet layer[40]. So, if security services are provided at the network layer, the basic IP software must be replaced by or enhanced with modules to add security mechanisms. Therefore, existing applications can make use of **IPSec**[41] to provide security at the IP layer in a transparent way. This means that we do not need to develop new secure applications. But Internet layer security has some disadvantages. One of them is, that it requires the use and deployement of new **TCP/IP modules**. This requirement can not be met by less widely operating systems, since it leads to a very costly process.

IP protocol

IPSec

TCP/IP modules

40 See chapter 2.

41 See chapter 3.

2. The use of transport layer security:
   SSL As mentioned in Chapter 3, **SSL** was originally conceived to protect the HTTP protocol. Thereafter, it was adapted to be able to provide security services for arbitrary TCP/IP based application protocols such as HTTP, FTP etc. One advantage of SSL is that transport layer security operates on an end-to-end basis. Since SSL is located between the transport and the application layer (see Fig. 7.1-1) each service of the application layer can directly use SSL-based connections and specify the required security mechanisms. This **Flexibility** **flexibility** is desired in case of HTTP sessions, because not all Web sites contain confidential information which require secure connections over the Internet.

   Traffic analysis Nevertheless, it is important to note that SSL does not protect against **traffic analysis**. For example, by examining the unencrypted source and destination IP addresses and TCP port numbers, or examining the volume of transmitted data, a traffic analyst can still determine what parties are interacting, what type of services are being used, and sometimes even recover information about business or personal relationships [Opp00].

   SSH Like SSL, the **SSH** protocol operates at the transport layer. But it is more conceived to secure the remote terminal access and the file transfer application[42]. **TLS protocol** The **TLS protocol** is another example of this approach, it provides communication security services for TCP based application. In fact, TLS provides more security than SSL[43]. But TLS has some disadvantages, it can be actually applied only over TCP.

3. Integration of security mechanisms into the application layer:
   Providing security at the application layer is also the most flexible, because the scope and strength of the protection can be tailored to meet the specific needs of the application [Opp00]. With this approach, application protocols must be modified to provide security services as well as programs written for these protocols. Another disadvantage is that these new applications must be available to both communication parties. For example, S-HTTP (Secure Hypertext Transfer Protocol) is a security-enhanced HTTP protocol. A S-HTTP session requires the availability of an S-HTTP-capable client and an S-HTTP-capable server.

4. Security services above the application layer:
   Finally, it is also possible to provide security services above the applicaion layer. In this way, application data is prepared in such a way that it can both be transmitted transparently with existing application protocol implementations, **PGP** and provide the security services required. For example, the program **PGP** enables a secure sending of e-mails by means of encryption and/or signature mechanisms in a common email-program.

42 For further details, see chapter 3.

43 See chapter 3.3

Which of the above listed possibilities is convenient to provide security services for TCP/IP based connections? Generally, we can not claim that one possibility is better than another one to provide security services. But the best choice depends on the services required and the application environment in which the services must be deployed [Opp00]. It is often the case that the security mechanisms are used not only in one layer but rather in multiple layers. This enables to combine all advantages of the possibilities to secure connections listed above.

There are many application protocols and services layered on top of TCP and UDP (User Datagram Protocol). The most important of them will be outlined next. For each application protocol, we will focus on its security weaknesses and give alternatives, or security-enhancements proposed to provide the security required. So, we will deal with:

1. Remote terminal access, implemented by the Telnet remote login protocol.
2. File transfer, implemented by the file transfer protocol (FTP).
3. The network file system (NFS) which uses RPC (Remote Procedure Call) to provide transparent file access over a network.

The topics S/MIME and executable content which are subject to network security, will be introduced in this chapter, too.

7.2 Remote Terminal Access

Telnet

Telnet is the standard for remote terminal access on the Internet [ZCC00]. In a Telnet session, the user initiates a connection to a remote host and client and server negotiate the connection. The user enters his user ID and password. If these steps have been completed successfully, the terminal session starts. From now on, the Telnet session looks like any other terminal session with a remote host. At the end of the session, usually the user has to exit the Telnet client manually. Depending on the configuration and implementation the client may also close the connection automatically.

Sniffing attack
Hijacking attack

Telnet was considered a fairly secure service because it is based on authentication. Unfortunately, Telnet sends all of its information unencrypted, which makes it extremely vulnerable to **sniffing** and **hijacking attacks**. With a sniffing attack, an attacker can read every network datagram sent over the network during the session. In a hijacking attack, an attacker is able to steal the Telnet session for his purposes. For example, if the authenticated user has logged to a remote host as root, an attacker mounting the hijacking attack can change the superuser password. For this reason, Telnet is now considered as one of the most unsecure services when used to access remote systems.

In fact, the basic Telnet mechanism is subject to all security considerations, namely authentication, access control, confidentiality and integrity [Hug95]:

User-to-host authentication

1. Telnet lacks a secure **user-to-host authentication**[44]. Note that Telnet users must supply login passwords.

2. Without the existence of external filtering tools, a Telnet server initiates connections to both trusted and untrusted networks.

3. Telnet sessions are subject to network eavesdropping, giving their users no confidentiality.

4. An untrusted router can alter the keystrokes sent by a user to a host, as well as the host's response back to the user.

Telnet would only be safe if the remote machine and all networks in between would be secured. This means that Telnet should not be used across the Internet.

VPN

To realize authenticated remote access, we make use of SSH[45]. SSH enables to securely log into a remote machine, to execute commands on that machine, and to transfer files from one machine to another. Another possibility to provide secure remote access is to create an encrypted network connection (a **virtual private network VPN**) and run normal Telnet across it.

Other security-enhanced Telnet software packages have been developed [Opp00] such as:

1. The `secure Telnet` software replacement for 4.4 BSD UNIX developed at AT&T Bell Laboratories.

2. The `secure RPC authentication` (`SRA`) software package was developed at Texas AT&T Bell Laboratories. However, the parameter of the secure RPC were successfully cryptoanalyzed. After this the authors of SRA have designed and implemented the NATAS protocol. NATAS-based software was then developed for Telnet and FTP.

3. `Secure Telnet` (`STEL`) is another secure Telnet software package for UNIX systems that was developed at the University of Milan in cooperation with the Italian CERT.

7.3 File Transfer

FTP

File Transfer Protocol (`FTP`) [PR85] is a simple client-server protocol FTP which provides a general-purpose mechanism for client manipulation of a server's file system. Using an FTP application, clients are capable to transfer files to and from the server, to obtain directory listings, to create directories, and to delete or rename files.

44 User-to-host authentication schemes identify users to computer systems. The purpose of this type of authentication is to provide users with services for which they are authorized, and to deny access to services for which they are not.

45 See chapter 3.3.

Web browsers support FTP as well as HTTP and will automatically use FTP to access URLs which start with ”ftp” such as ”ftp.fernuni-hagen. de”.

An FTP site can be public, private, or both. With a private account, a user can be given access to the entire network’s directory structure, or just specific areas. The Internet is also home to thousands of public access servers that allow anyone to connect and transfer files to and from specific directories regardless of whether they have an account on the host. This is called **anonymous FTP**. In order to access an anonymous FTP site, a user usually log in with ”anonymous” as the user name and ”guest” or the email address as password [KDS+00].

Anonymous FTP

FTP still has its own security weaknesses:

1. FTP lacks a secure user-to-host identification. Authenticated (non-anonymous) FTP users provide login passwords in cleartext.
2. Unless external filters are applied, an FTP server can connect to both trusted and untrusted networks.
3. FTP sessions are subject to network eavesdropping and therewith no confidentiality can be provided.
4. An untrusted host can alter data sent between the client and the server.

Another security risk which affects anonymous FTP servers is that unauthorized users can get access to private areas or files, and can use FTP to get **shell-level access** to the system itself.

Shell-level access

Secure file transfer can be offered with the use of SSH. Note that the **security-enhanced FTP** is yet not of great importance in practice.

Security-enhanced FTP

Exercise 7.3-1:

Explain why remote terminal access with Telnet and file transfer with FTP are not secure? How can we secure these two applications?

7.4 Network File Systems

Protocols for **file sharing** allow computers to use files that are physically located on disks of other computers (see Fig. 7.4-1). File sharing protocols can be used as file transfer protocol. Therefore, the file must be shared first and then copied locally. File sharing protocols are much more complicated to implement than file transfer protocols. File sharing protocols provide **transparency** and **rich access**. Transparency means that the file appears to be local and the user does not see the file sharing occurring. With rich access, the user can read and/or write and/or execute the remote file. On the other hand, transparency provided from the file sharing protocol puts limits to the sort of security to be implemented, and the need to provide rich access makes file sharing protocols more complex to implement [ZCC00].

File sharing

Transparency

Rich access

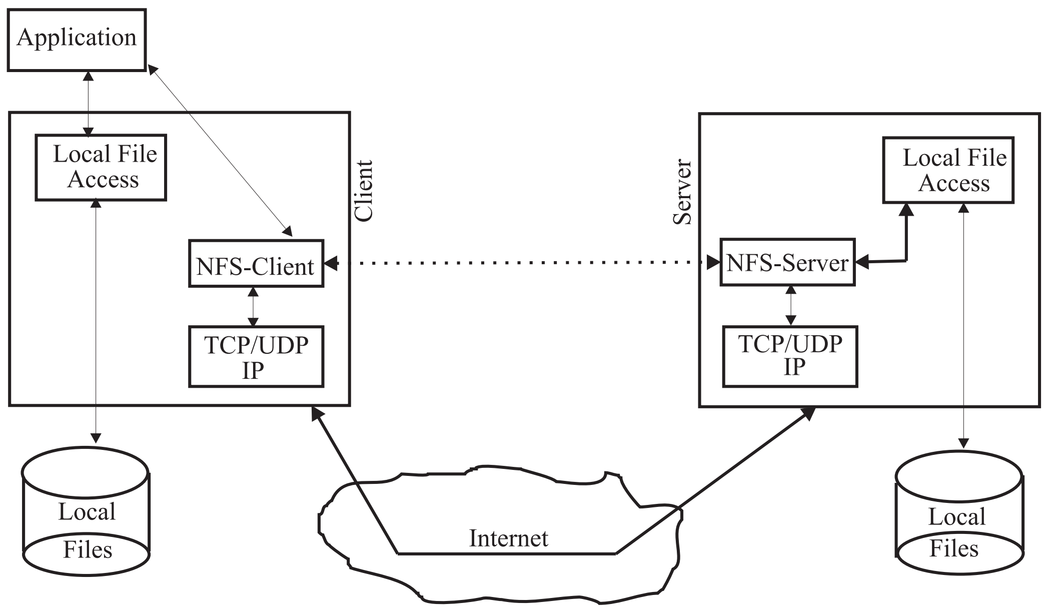


Fig. 7.4-1: File sharing between NFS client and server [FRU00].

The most commonly used file sharing protocols are the `Network File System` (**NFS**) based on UNIX, the Common Internet File System (CIFS) under Microsoft Windows and Appleshare on Apple Macintosh computers. **NFS**

Network File system (NFS)

NFS is implemented on different machine types, operating systems, and network architectures. This platform independence is made possible by the use of the `Remote Procedure Calls` (**RPC**) [SM88]. **RPC**

RPCs allow the client process to direct the server process to execute procedure calls as if the client process had executed the calls in its own address space [N98]. Note, that the client and the server are two separate processes. Therewith, they do not need to exist on the same physical system (although they can). Since it is possible that the two systems do not represent data in the same manner, RPC uses data types defined by the `eXternal Data Representation` (**XDR**) protocol [SM87] to hide all details of architecture-dependent data representation[46]. **XDR**

46 XDR is the specification for a standard representation of various data types. By using a standard data type representation, a program can be confident that it is interpreting data correctly, even if the source of the data is a machine with a completely different architecture. In practice, most programs use the data type representation specific to the architecture of the computer on which they are running. When the program needs to communicate with another program (which runs on another computer), it converts its data into XDR format before sending the data. Conversely, when it receives data, it converts the data from XDR format into its own specific data type representation [N98].

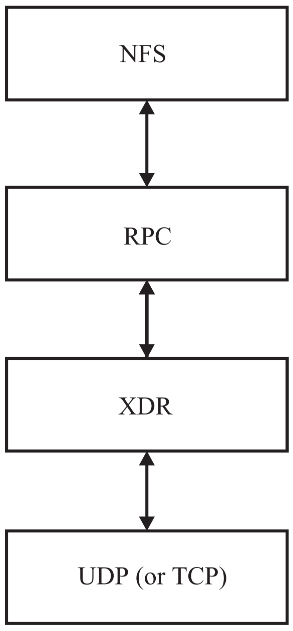


Fig. 7.4-2: NFS abstraction with RPC and XDR [Hug95].

From the security perspective, the NFS[47] works in an unsecure manner. The request from a client to a server as well as the content data will be transmitted using the TCP or UDP protocol. In other words, the confidentiality will be compromised without the implementation of **encoding mechanisms** in the used (unsecure) transport protocol. On the other hand, an access control to files and directories must be assigned in both local and distributed file systems. Therewith, only authorized users will be able to read or manipulate files. Unfortunately, the last requirement is not adequately met in NFS.

Encoding mechanisms

As mentioned above, NFS uses RPC to provide transparent file access over a network. The RPC possesses a mechanism for authenticating clients and servers. In fact, there are several mechanism for authenticating clients that apply to NFS, two of them are **AUTH_UNIX** and **AUTH_DES**. The `AUTH_UNIX` authentication is used by standard NFS implementations and provides no real form of authentication. In fact, this authentication is employed in environments where very minimal security is acceptable and a high level of distributed file system performance is required.

AUTH_UNIX
AUTH_DES

In the `AUTH_UNIX` authentication, only the client credentials including a host name, UID, GID and adjunct GIDs are sent across the network. These credentials act as means for authentication. NFS also has a very weak **client authentication**. Using this security weakness, an attacker can convince the NFS server that a request is coming from a client that is permitted in the exports file. Note, that in the **export file** (`/etc/export`) of the server, if it is properly configured, there will be specified which file systems can be mounted and which machines can mount them.

Client authentication

Export file

If the NFS uses the `AUTH_DES` authentication, a higher level of security (in comparison to the `AUTH_UNIX` authentication) can be reached. In the `AUTH_DES` authen-

47 We consider NFS version 2, see [SM89].

Diffie-Hellman algorithm

tication, the RPC client and server use the **Diffie-Hellman algorithm**[48] to derive a DES session key, which is used to encrypt communications for the duration of the client's login session. Even if the system provides the `AUTH_DES` authentication, it can not provide the security desired, since [Hug95]:

1. The one-pass DES is already decrypted.
2. The modulo in the Diffie-Hellman algorithm must be adequately chosen. Otherwise, the communication for the duration of the client user's login session could be easily decrypted.

Andrew File System AFS

AFS

Kerberos

AFS is a security-enhanced distributed file system. It was originally developed at Carnegie Mellon University and is marketed since 1997. Comparable to NFS, AFS provides a considerably higher level of security. It uses the **Kerberos**[49] system, an authentication and distribution key management system developed by the researchers at MIT (Massachuserts Institute of Technology) [FRU00]. Kerberos offers additional layers of protection beyond what is available with simple private or public key based services. Kerberos introduces the notion of a ticket which is a credential containing expanded information about the user, and an encrypted DES key assigned by a trusted host called the Key Distribution Center (KDC). The DES key is safely distributed to both client and server, ensuring mutual authentication in much the same way as the scheme using Diffie-Hellman.

Authorization service

We will now make a comparison between **authorization service** provided by NFS and AFS. The purpose of an authorization service is to ensure that users have the permission to do what they are attempting to do once they have been successfully authenticated. The NFS employ the **UNIX style authorization** in which both client and server must either share the same ID list, or there must be a mapping between the two [Hug95]. Once the mapping is established, users are authorized to manipulate files. In addition to UNIX Modes authorization, AFS uses special

ACL

`Access Control Lists` (**ACL**) in order to determine whether and how a user may access files [FRU00].

Since Kerberos and AFS require significant technical expertise to set up and maintain, AFS is not widely used outside of a relatively small number of sites.

Cryptographic File System (CFS)

CFS

The **Cryptographic File System** (`CFS`) is another secure distributed file system [FRU00]. The CFS pushes encryption services into the UNIX file system. The task of CFS is to encrypt files when

1. they are stored on a local harddisk,

48 See [KCL+00].

49 Kerberos Version 4.

2. they are sent to a remote file server,
3. they are stored on a remote file server, or
4. backups of them are generated.

Users associate a cryptographic key with the directories they wish to protect. Files in these directories (as well as their pathname components) are transparently encrypted and decrypted with the specified key without further user intervention. CFS employs a combination of various DES-modi to provide high security with good performance on a modern workstation [FRU00]. CFS can use any available file system for its underlying storage without modification, including remote file servers such as NFS.

7.5 Secure Electronic Mail With S/MIME

Electronic mail (or email for short) is a widely used, easy, and fast communication service provided on many different kinds of computers. It is mostly used to transfer text messages from one user to one or more recipients, but also multimedia content is supported. Like regular mail, electronic mail is a **point-to-point communication**. One person originates the communication by composing a message and then sending it to one or more recipients. The message is delivered to the mailbox of each recipient, where it can be read.

Point-to-point communication

Electronic mail has become one of the most important Internet services during the recent years. This fast asynchronous messaging service is used by millions of users each day, both in private and commercial environments. Today, most emails are transmitted over the Internet absolutely unprotected. An examplary list of things an adversary can do with emails may look as follows:

- read arbitrary emails
- change the content of an email
- change name of sender and receiver
- intercept and delete the email

Email security aims to make the above mentioned attacks much harder for a potential adversary. The first three items can be prohibited by applying standard cryptographic primitives: encryption and digital signature. It must be mentioned though, that cryptographic means do not suffice to prevent an adversary from carrying out denial of service attacks as represented by the last item in the list above.

Currently, two major standards for email security exist: S/MIME and OpenPGP. The PKI related features of the OpenPGP standard have already been described in chapter 7 of the course `Applications of Cryptology`. Additionally, OpenPGP also defines a message format for email encryption. In contrast to OpenPGP, which uses its own proprietary certficate and message formats, S/MIME widely employs existing standards like X.509 and PKIX. For this reason, it is broadly supported by industry leaders like Microsoft, Netscape, Novell, and Lotus. Due to the limited scope, only S/MIME will be described in detail in this chapter. For information on the message format of OpenPGP, please refer to RFC 2440 [ea98].

In the following section, first a thorough introduction to the structure of email messages is given. The basics will be essential to understand the security related features that are provided by S/MIME.

7.5.1 Structure Of Email Messages

Basically, electronic mail involves transferring a computer file from one user account to another user account. In the following we will use the paradigm of paper mail to explain the way electronic mail works. In Fig. 7.5-1 the two kinds of mail can be seen side by side.

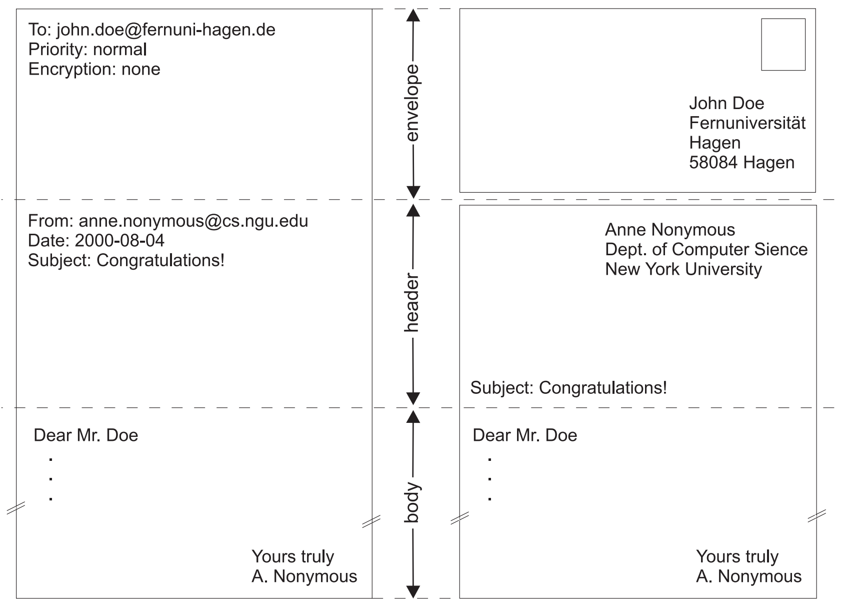


Fig. 7.5-1: Traditional mail vs. email.

The equivalence between the combination of the labeled envelope and the head of the letter on one side and the header of the email on the other side can clearly be seen. It is analogous to the matching of postal and email body.

The structure of email messages was defined some years ago. The RFC 822 [Cro82], the de facto Internet standard) shows the definition in detail and is used as basic literature for this section.

Message envelope
Header
Message body

The email message has three basic parts: The **message envelope**, the **header** and the `message body`. The message header lists information about the sender, the recipient, the posting date, the subject of the message, etc. The **message body** contains the actual message being sent. The message header consists of several fields. Take a look at the following example of an email message (all data of the email is shown; an email client will suppress some of these while displaying, some more fields can be displayed in other situations)

Example 7.5-1:

```
Received: from SpoolDir by LGKS1 (Mercury 1.31);
          10 Jul 100 16:22:48 MET
Return-path: <anne.nonymous@cs.nyu.edu>
Received: from elm.fernuni-hagen.de by lgks1.fernuni-hagen.de
          (Mercury 1.31) with ESM11 Jul 100 01:41:00 MET
Received: from slinky.cs.nyu.edu by elm.fernuni-hagen.de via
          Internet
with ESMTP; Tue, 11 Jul 2000 01:37:50 +0200
Received: from localhost (anne.nonymous@localhost)
by slinky.cs.nyu.edu (8.9.1/8.9.1) with ESMTP id TAA27290
for <Thomas.Demuth@FernUni-Hagen.de>;
Mon, 10 Jul 2000 19:37:48 -0400 (EDT)
From: Anne Nonymous <anne.nonymous@cs.nyu.edu>
To: thomas.demuth@FernUni-Hagen.de
Date: Mon, 10 Jul 2000 16:21:33 +0200
MIME-Version: 1.0
Content-type: text/plain; charset=ISO-8859-1
Content-transfer-encoding: Quoted-printable
Subject: Thanks
Priority: normal
X-mailer: Pegasus Mail for Win32 (v3.12a)
X-PMFLAGS: 34078848 0 1 Y0FC54.CNM

Dear Mr. Demuth,
thank you for sending the research report.
Regards,
Anne Nonymous
```

The header of this message consists of the fields from the first ”Received” to ”X-PMFLAGS”, the body starts with ”Dear Mr. Demuth,”. Header and body must be separated by a single line. Headers also pick up information as a message is sent to its recipient. The email software adds hidden text that itemizes the messages size, origination location, and other miscellaneous information. By the time the message arrives at its destination, it will have a whole paragraph of additional text that recounts the routers the message passed through.

7.5.2 The Email Header

The header consists of ”fields”. Each field has a name (and therefore a meaning), a value, and is terminated by a newline/line feed. The field name must be composed of printable **ASCII characters**, the body may be composed of printable ASCII characters including newline/line feed. The header fields The most important and often used header fields are explained in the following. Details can be found in RFC 822. The minimum required fields are ”Date”, ”From”, and ”Bcc” or ”To”; all others are optional.

ASCII characters

To: This field contains the mailing addresses to which the message is to be sent. If more than one address is listed, they must be separated by commas.

Subject: The contents of the ”Subject” field should be a piece of text that says what the message is about. The reason ”Subject” fields are useful is that most mail-reading programs can provide a summary of messages, listing the subject of each message but not its text.

Cc: This field (”Carbon Copy”) contains additional mailing addresses to send the message to. This works like ”To”’ except that these readers should not regard the message as directed to them.

Bcc: This field (”Blind Carbon Copy”) contains additional mailing addresses to send the message to, which should not appear in the header of the message actually sent. Copies sent this way are called blind carbon copies.

From: This field is used to specify the sender, when the account he is using to send the mail is not his own. The contents of the ”From” field should be a valid mailing address, since replies will normally go there.

Received: The names of the sending and receiving hosts and time-of-receipt may be specified in this field. The ”via” parameter may be used to indicate what physical mechanism the message was sent over and the ”with” parameter may be used to indicate the mail- or connection-level protocol that was used, such as the SMTP mail protocol, or X.25 transport protocol. Some transport services queue mail; the internal message identifier that is assigned to the message may be noted, using the ”id” parameter. When the sending host uses a destination address specification that the receiving host reinterprets, by expansion or transformation, the receiving host may wish to record the original specification, using the ”for” parameter. For example, when a copy of a mail is sent to a member of a distribution list, this parameter may be used to record the original address that was used to specify the list.

Reply-to: This field directs replies to a different address. Most mail clients automatically send replies to the ”Reply-to” address in preference to the ”From” address.

In-reply-to: This field contains a piece of text describing a message for replying to. Some mail systems can use this information to correlate related pieces of mail.

References (Message Threading): This field is usually maintained and added automatically. ”References” describes the thread (chain, series) of messages that ”came before” the actual one. Each message on the Internet has a unique identification number that is kept in the ”Message-ID” field. ”References” is a list of the message-ids from previous messages in this thread. When a new message is composed and sent, the mail server adds the ”Message-ID” field.

When someone replies to that message, the user’s email program will create a ”References” field and copy the original message-id into it. If someone else replies to that reply, this ”References” field gets the previous references, plus the previous message-id. And so on. An example should make this easier to see. Let’s look at a thread with three messages. user-A sends a message to user-B. This original message header might have:

```
From: user-A
To: user-B
Subject: Something very important
Message-ID: <123@fernuni-hagen.de>
```

Now user-B replies to user-A. The reply header looks like:

```
From: user-B
To: user-A
Subject: Re: Something very important
References: <123@fernuni-hagen.de>
Message-ID: <246@fernuni-hagen.de>
```

The third message in this thread is from user-A to user-B, with a copy to user-C. user-A also decides to change the subject:

```
From: user-A
To: user-B
Cc: user-C
Subject: Another thing (was: Re: Something very important)
References: <123@fernuni-hagen.de> <246@fernuni-hagen.de>
Message-ID: <789@fernuni-hagen.de>
```

The ”References” field will keep getting longer as this thread goes on. It lets one find all the messages in the thread by searching for the message-ids.

X- (Own Header Fields): The RFC 822 transport standard carefully specifies what header fields are legal in mail messages. Any field name starting with the characters X- (the letter X followed by a dash) will not be adopted by a new message header standard. According to RFC 822, ”any field which is defined in a document published as a formal extension to this specification [will not] will have names beginning with the string ‘X-’.” That means the standard cannot anticipate every useful header, it allows for user-defined fields as extensions to the standard. The convention is to begin a user-defined field with ”X-”, because the standard guarantees that no official extension will use a header with those names. So the X-headers can be seen as a free ”playground” for anyone’s use (see Example 7.5-2).

Example 7.5-2:
The web-based email service Hotmail includes an `X-Originating-IP` header which holds the IP address of the sending computer. This can serve as a means to identify the sending computer although the Hotmail email account is anonymous.

Structure of an email address

Email messages employ a two-part addressing scheme. A valid Internet email address is: `thomas.demuth@fernuni-hagen.de`
Email has its own adressing system, called domain name addressing (corresponding to the domain name system of the Internet as the electronic equivalent of postal addresses. Like these postal addressses, email addresses go from specific to general, in order to route the message to the right computer and mailbox (or person). Generally, email addresses have a username, one or more location identifiers, and a domain. An ”@” (”at sign”) symbol separates the user name from locations and domain.

The email body and MIME

The mail body is completely free of format, one can compose an email in any desired way. Corresponding to the standard, the body is separated by a blank line from the header. In the early days of computers email was not ”8 bit” clean; users were able only to write emails in straight (7 bit) ASCII text. That means that only textual email is supported by RFC 822 email; nontextual byte content can therefore be clipped to seven bits.

To solve this problem, in 1992 the MIME standard was developed. MIME stands for ”Multipurpose Internet Mail Extensions” and enables email to transport multimedia data, still being conform to RFC 822.

Before MIME, users converted their binary files like spreadsheets or other office documents with the (UNIX) commands ”uuencode” and ”uudecode”. These commands are used to transmit binary files over transmission mediums that do not support other than simple ASCII data. Uuencode reads the input and writes an encoded version. The encoding uses only printable ASCII characters and includes the mode of the file and the operand name for use by uudecode.

But, since the introduction of MIME no user has to care about this any more while writing emails (As a consequence of this, one can attach any binary file to an email.).

The following RFCs define Multipurpose Internet Mail Extensions, replacing the RFC 1521 which is not valid anymore:

- RFC 2045 [ea96c]: MIME Part One: Format of Internet Message Bodies
- RFC 2046 [ea96d]: MIME Part Two: Media Types

- RFC 2047 [Moo96]: MIME Part Three: Message Header Extensions for Non-ASCII Text
- RFC 2048 [ea96b]: MIME Part Four: Registration Procedures
- RFC 2049 [ea96a]: MIME Part Five: Conformance Criteria and Examples

These standards (or definitions) redefine the simple structure of messages in RFC 822 format to allow for textual message bodies in character sets other than US-ASCII, an extensible set of different formats for nontextual message bodies, multi-part message bodies, and textual header information in character sets other than US-ASCII. It is important to understand, that MIME does not change the structure of an email. In fact it encodes the arbitrary data in ASCII to be transmitted in a standard email message. Therefore, older email programs (email clients) can handle MIME messages, too.

To accomodate arbitrary data types and representations, each MIME message includes information that tells the recipient the type of the data and the encoding used. MIME information resides in the RFC 822 mail header - the MIME header files specify the version of MIME used, the type of the data being sent, and the encoding used to convert the data to ASCII.

Above, the possible encoding of binary messages or attachments with uuencode/base64 has been mentioned. The advantage of MIME is a quite more comfortable handling for the user. In addition to the defintion of encoding and formatting, MIME does also determine the methods modern email clients can detect and handle such messages autonomously, i.e. starting the appropriate application. For this reason, MIME messages contain meta information, defining the player for a sound file or a program to work with graphics.

MIME is a flexible way to embedd multimedia attachments in Internet messages. Data can be nested and instructions to a front end program can be embedded to output the information in dependence of available resources. In this way, a memo can be read out or displayed alternatively as text, if no sound device is available. Furthermore, links to external data can be integrated in MIME messages.

Example 7.5-3 illustrates a MIME message that contains a picture in standard GIF represenation. The GIF image has been converted to a 7-bit ASCII representation using the base64 encoding (for explanation of base64, see below).

Example 7.5-3:

```
From: Anne Nonymous <anne.nonymous@cs.nyu.edu>
To: thomas.demuth@FernUni-Hagen.de
Subject: Congratulations
Date: Mon, 10 Jul 2000 16:21:33 +0200
MIME-Version: 1.0
Priority: normal
X-mailer: Pegasus Mail for Win32 (v3.12a)
```

```
X-PMFLAGS: 34078848 0 1 Y0FC54.CNM
Content-type: Image/GIF; name="medal_of_honour.gif"
Content-disposition: attachment; filename="medal_of_honour.gif"
Content-transfer-encoding: BASE64

R0lGODdhWAK+AKIAAP///9vb27a2tpKSkm1tbUlJSSQkJAAAACwAAAAAWAK+AEAD
/wi63P4wykmrvTjrzbv/YCiOZGmeaKo6B7G+8BkchtI2Rh3vfJf3wGCpoBMaj47B
weaSDBiHaOO2MDxthqj2SVgClBMaoLssLGcDwSHAPghsEbVCPZgFFnIW6VBodB9/
ClkAWmsLXohSfHheDHk2V1MGbF1qdwp2iG8/bmFvbUuGhGIEOmAaREiqqxOpDqlm
UUUABn1TWm+ljo0HaY0zOLYRVg9KwoRvCgNZTW7AhwNmcDaNh8kXp1RO1Vm9h7OH
AcZfoZd/1QAzVOiCxwynDWpS4YQPP9MjzH7s8Ay6tE1o1RC1QFqEagTfgZsTytaz
BWjSeckTy8VDfCQCsYLirv8ToQIzatV4dqfLMVGXNkb418AVAyveIElwM4jWvDUG
Ak4rYCtmAQKxttSDxqKWGR2i7o0iojPdiEA1ZUhgI4ANnpTGCFwbRaOW
.
.
.
```

The MIME field `MIME-Version 1.0` declares that the message was composed using version 1.0 of the MIME protocol. MIME is downwards compatible, that means, that every MIME capable email client can likewise handle ASCII messages.

Two more fields (not used in the example email) are `Content-Id`, acting as a non-ambiguous identifier (and functionally identical to the standard header `Message-ID` (see above)) and `Content-Description`, a text describing the content, understandable for a human reader to inform him before the content is decoded. Thus the receiver of the message can decide if it is reasonable for him to decode the message.

The `Content-Transfer-Encoding` header declares in the example, that base64 encoding was used to convert the image to ASCII. There are five different encoding schemes available (and another one for further extensions):

7bit: ASCII text, using 7 bit characters, to directly transfer the content without further encoding. This format is limited to 1.000 characters per line.

8bit: ASCII text, using 8 bit per character (values from 0 to 255) and thus violating the encoding scheme of RFC 822. Nevertheless it is used, causing the expected consequences.

binary: Like ”7bit” without the limitation to 1.000 characters per line, but with the consequences of the ”7bit” mode.

base64: This encoding is also called ”ASCII armor”. Groups of 24 bits are encoded in four units of six bit. Each of these units is then encoded as regular ASCII character (”A” for 0, ”B” for 1, and so on). Carriage returns and line feeds are ignored.

quoted-printable: This encoding only affects bytes exceeding the limit of the seven-bit-ASCII-code. They are transformed into an equal sign, followed by the hex code of the character (for example: ”=E9” for an ”¨a”). This type of encoding is useful, if special, national characters are used, or if it is not known, if the receiver of a message can handle base64-encoded attachments.

The `Content-Type` declaration specifies the type of information. Seven types are defined, each with one or more subtypes; types and subtypes are separated with a slash (”/”). The purpose of the `Content-Type` field is to describe the data contained in the body in such a manner that the receiving user agent can pick an appropriate agent or mechanism to present the data to the user, or otherwise deal with the data in an appropriate manner. The value in this field is called ”media type”.

Table 7.5-1 shows the most often used MIME types and subtypes.

Tab. 7.5-1: *MIME types and Subtypes.*

Type	Subtype	Description
text	plain	unformatted ASCII text
	richtext	basically formatted ASCII text
image	gif	GIF picture
	jpeg	JPEG/JPG picture
audio	basic	sound data
video	mpeg	MPEG movie
application	octet-stream	non interpretable byte sequence
	postscript	printable postscript document
	rfc822	MIME message in RFC 822 format
message	partial	message has been split before transfer
	external-body	link to external ressource
multipart	alternative	same message in different formats
	parallel	the parts of the message have to be displayed at once
	digest	each part is a RFC 822 compliant message
	mixed	several parts of independent encoding

The understanding of MIME is not only important to realise the structure and interpretation of email messages. The Hypertext Transport Protocol (HTTP), used to transfer Web pages, does also use MIME for the transport of the pages.

It is the task of the sender’s email client to interprete the data to send and to select the right content and sub types. In some cases the sender has to interfere and select the types manually.

In Example 7.5-3 the `Content-Type` declaration specifies that the data is a GIF image, image is the content type, and gif is the subtype. To view the image, a receivers email client must first convert from base64 encoding back to binary, and then run an application that displays a GIF image on the users screen.

In Example 7.5-3 an additional MIME header field can be seen: `Content-disposition`. This field gives more information about the following MIME element. Like in the case of the `Content-type` field further parameters can be added to the line. In the example one can see the name of the image and that it is an attached element.

Multipart MIME messages

One very important type mentioned above is `multipart`. This type allows messages consisting of several parts; these parts are clearly separated from each other. With the multipart content type, email gets considerable flexibility. The subtype `mixed` allows a single message to contain multiple, independent submessages, each one having its own content type and encoding. Mixed messages make it possible to include different multimedia data in one message. The subtype `alternative` allows a single message to include multiple representations of the same data. This approach is usefull for sending the same message in different formats (audio, video, or text). The email client of the receiver should be able to decide the part of message which will be used, with respect to the hardware capabilities of the computer. Subtype `parallel` permits a single message to include subparts that should be used together (like separate audio and video data that must be played synchronously). Subtype `digest` allows a single message to contain a set of other messages that are RFC 822 compliant. This feature is very useful for receiving the daily or weekly email of a mailing list at once. The following example shows a multipart mixed MIME message

Example 7.5-4:

```
Date: Fri, 21 Jul 2000 14:02:28 +0200
From: Anne Nonymous <anne.nonymous@cs.nyu.edu>
Organization: MAD Dept.- Department for Mathematics and
          Depressions
X-Mailer: Mozilla 4.7 [en] (Win98; U)
MIME-Version: 1.0
To: thomas.demuth@fernuni-hagen.de
Subject: Betreff
Content-Type: multipart/mixed;
boundary="------------675454AB4619496261BDECC9"

--------------675454AB4619496261BDECC9
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

This is plain text (see Content-Type)
--------------675454AB4619496261BDECC9
Content-Type: image/jpeg;name="et-online.jpg"
Content-Transfer-Encoding: base64
Content-Disposition: inline;filename="et-online.jpg"

/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAsICAoIBwsKCQoNDAsNERwSEQ8PE
SIZGhQcKSQr
.
.
```

```
Y9FFFbCCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigA
ooooAKKKKACiiigAooooAKKKKAHdqQ8dDRRSA//2Q==
--------------675454AB4619496261BDECC9
Content-Type: application/x-unknown-content-type-Winamp.File;
name="Elvis Presley - Jailhouse Rock.mp3"
Content-Transfer-Encoding: base64
Content-Disposition: inline;filename="Elvis Presley - Jailhouse
Rock.mp3"

//uQRAAAAAAASwAAAAAAAAlgAAAAAAABLAAAAAAAACWAAAAA//////////////
//////////
.
.
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAER
--------------675454AB4619496261BDECC9--
```

7.5.3 Securing MIME-based Emails With S/MIME

In the following, we will describe the procedure that Alice and Bob have to go through if Alice wants to send an encrypted an signed email to Bob:

1. Alice generates a `content-encryption key` (session key).
2. Alice encrypts the message with the `content-encryption key`.
3. Alice encrypts the `content-encryption key` with Bob's public key (encryption key). Alice may receive Bob's public key in form of a certificate.
4. Alice concatenates the encrypted message with the encrypted `content-encryption key`. Alice may append further information like the utilized algorithms to the message. The construct consisting of the message, the `content-encryption key`, and further information is also called `token` or `digital envelope`. If Alice only seeks privacy, she can directly send the `envelope` to Bob.
5. If Alice also wants data integrity and message origin authentication she has to digitally sign the `envelope`. The signature is carried out with Alice's secret signing key.

This basic procedure can be found in every email encryption product, both OpenPGP or S/MIME based. Nevertheless, this protocol only applies to simple mails which do not consist of several parts. In a multipart MIME email, all parts have to be signed and encrypted separately.

It is obvious, that to allow interoperation between diverse email clients the rather vaguely described operations and message format in the procedure above have to be exactly specified. The two most important issues which must be agreed on are:

1. the certificate handling, that is the utilized PKI, and

2. the `envelope` or message format.

In the case of S/MIME Version 3 this is done in two RFC documents:

- RFC 2632 : ”S/MIME Version 3 Certificate Handling” [Ram99a].
- RFC 2633 : ”S/MIME Version 3 Message Specification” [Ram99b].

Cryptographic Message Syntax

The RFC 2632 specifies that a hierarchical X.509 PKI as defined by the PKIX working group `must` be utilized (see the course `Applications of Cryptology`). The ”Message Specification” is based on another IETF standard, namely the **Cryptographic Message Syntax** (`CMS`), which is specified in RFC 2630 [Hou99]. CMS defines how an arbitrary message can be cryptographically signed and encrypted. RFC 2633 specifies how the CMS can be applied to MIME messages.

CMS is derived from the RSA de-facto standard PKCS #7 (RFC 2315) and specifies the following:

- Content type for signed data.
- Content type for digital envelopes (as described above).
- Digested-data content type. This normally serves as input for the enveloped-data content type. This data type holds the message digest of any type of content and thus provides content integrity.
- Encrypted-data content type. This differs from the envelope type in that no session keys and further information is regarded.
- Authenticated-data content type. This type consists of arbitrary content, a message authentication code (MAC), and encrypted authentication keys for one or more recipients. The authentication key is randomly generated by the orignator of the message and encrypted with the public key of the recipient. The MAC is constructed by encrypting the message with the authentication key.
- Supported algorithms. The cryptographic algorithms that must and should be supported, e.g. CMS implementations must include DSA and may include RSA.

S/MIME Version 3 defines how to create a cryptographically enhanced MIME body part by employing CMS. S/MIME can be regarded as an CMS application and provides the following cryptographic security services for electronic messaging applications: authentication, message integrity and non-repudiation of origin[50] (using digital signatures) and privacy and data security (using encryption).

S/MIME messages are a combination of MIME bodies and CMS objects. To improve interoperability, S/MIME rigidly defines which options of CMS must be supported. Each MIME entity must be prepared for signing and/or enveloping. To

50 Non repudiation of origin prevent the originator of a message denying that he has originated the message.

do this, the MIME entity is canonicalized and encoded for transfer. The task of **canonicalization** of a MIME entity is to render it into a standard form that is invariant as it goes through various email components. In fact, the exact details of canonicalization depend on the actual MIME type and subtype of an entity. For example, canonicalization of type `text/plain` is different from canonicalization of `audio/basic`. The canonical MIME entity is fed into a CMS processing facility which produces a CMS object. The CMS object is then finally wrapped in MIME. It must be noted, that the RFC 822 [Cro82] headers of an email are not included in S/MIME.

Canonicalization

S/MIME defines the `application/pkcs7-mime` type. It also defines an optional ”smime-type” parameter which conveys information about the security applied (signed or enveloped).

Enveloped messages

An enveloped-only message is created as follows:

1. The MIME entity is canonicalized.
2. TheMIME entity is processed into a CMS object of type `envelopedData`. The envelope holds the encrypted message, the session key encrypted for the receivers, and the session key encrypted for the originator of the message.
3. The CMS object is inserted into a `application/pkcs7-mime` entity.

The file extension `.p7m` is given for this type of message.

Example 7.5-5: Enveloped-only data
In this example, ”enveloped-data” denotes the smime-type parameter for messages that are enveloped but not signed.

```
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H
f8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
0GhIGfHfQbnj756YT64V
```

Signed messages

There are two formats for signed messages defined by S/MIME: `application/pkcs7-mime` with `signedData` as parameter, and `multipart/signed`.

A signed message using the `application/pkcs7-mime` type is created as follows:

1. The MIME entity is canonicalized.
2. The MIME entity is processed into a CMS object.
3. The CMS object is inserted into a `application/pkcs7-mime` entity.

The file extension for this type of message is also `.p7m`. A receiver who wants to read the signed message, has to be capable of processing S/MIME entities.

Example 7.5-6: Signed-only data

```
Content-Type: application/pkcs7-mime; smime-type=signed-data;
name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

567GhIGfHfYT6ghyHhHUujpfyF4f8HHGTrfvhJhjH776tbB9HG4VQbnj7
77n8HHGT9HG4VQpfyF467GhIGfHfYT6rfvbnj756tbBghyHhHUujhJhjH
HUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H7n8HHGghyHh
6YT64V0GhIGfHfQbnj75
```

The advantage of the `multipart/signed` message type is, that a receiver must not necessarily be S/MIME capable to read the message. In this case the receiver cannot verify the signature but can at least read the message. This method is also referred to as clear-signing. The `multipart/signed` MIME type is specified in RFC 1847 [ea95]

. A `multipart/signed` message contains two parts: The first part is the body part over which the signature is created (including its MIME headers), the second part carries the control information which is necessary to verify the signature. The file extension for this type of message is `.p7s`.

Example 7.5-7: multipart/signed data

```
Content-Type: multipart/signed;
protocol="application/pkcs7-signature";
micalg=sha1; boundary=boundary42

--boundary42
Content-Type: text/plain
```

```
This is a clear-signed message.

--boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--boundary42--
```

Enveloped and signed messages

Finally, we have to know how to both sign and encrypt a message. This is achieved by nesting the signed only and eneveloped-only formats. The message can either be signed first and then encrypted, or enveloped first and then signed. This is possible because all the formats are MIME entities. When the message is first signed and then enveloped, the signatories are hidden. When the message is first enveloped and then signed the signatories are exposed, but the signatures can be verified without removing the enveloping.

A list of all documents relevant for S/MIME can be found on the WWW page of the respective IETF working group: http://www.imc.org/ietf-smime/

7.6 Executable Content

Executable content

This section deals with potential risks related to **executable content**. A user of a computer connected to the Internet has the possibility to download programs and execute them. This can be dangerous, since such programs can have **bugs** that may cause a computer to crash, other can be malicious and do damaging things, such as erasing a harddisk or transmitting confidential data to secret locations somewhere on the Internet. On the other side, an attacker aims to be able to execute malicious programs on a desired computer system without the knowledge of the correspon-ding victim. To do this in an easy way, the attacker can provide the victim with a program to execute. Note that the victim cannot determine what a program will do without running it. Moreover, it is impossible to determine what a program is doing while or after it is running, since some programs have the ability to hide their actual operations. Even operating systems with good security features do not offer sufficient protection against malicious programs that they download and execute locally. This is due to the fact that once the program is running, it inherits all the **privileges** and **access rights** from the user who invoked it.

Bugs

Privileges

Access rights

Well written executable content

In fact, executable content is not always malicious. **Well written executable content** makes Web pages more interesting and attractive. It enables Web pages to provide animations, interactive games, and other multimdia objects. The question now is how can we decide whether executable content is well written or not. The answer to this question is not evident.

From a theoretical point of view, the security problem related to executable content arises because there is no fundamental difference between a program (representing an active component) and data (representing a passive component) with regard to its internal representation within a computer system [Opp00]. Both data and programs are treated in the same manner and stored in the same memory. Consequently, a program is able to modify data, programs and eventually itself. This fact is used in **computer viruses** that are computer programs having the ability to manipulate and directly modify other program codes.

Computer viruses

In the following, we treat some executable content and their security risks. As an example of executable content, we focus on binary mail attachments, helper applications and plug-ins, scripting languages, Java applets, and ActiveX.

7.6.1 Binary Mail Attachments

Binary mail attachment

A **binary mail attachment** is an attachement to an email message that can encode an excutable program. The sender of a mail message attaches the desired binary data file(s) to the message. The receiver can automatically or manually execute the file(s) upon reception.

The following example demonstrates the dangerousness of a binary mail attachments if it is used for malicious tasks.

Example 7.6-1: ”I LOVE YOU VIRUS”[51]
”I LOVE YOU” is a macro virus detected in May 2000. A user will receive an email with ”I LOVE YOU” as the subject, and the email will have an attachment (a file) that contains the macro virus. ”I LOVE YOU” is a Visual Basic script macro. The virus is infectious when a user has a program (Microsoft Office or Microsoft Outlook) that can run it. If the user opens the attachment or saves it and then runs it, it will infect the user’s system. This virus is very destructive (if the user has Microsoft Office products installed). It does the following:

1. Searches the user’s computer for music, picture and html related files (.MP3, .JPG, .JS, .CSS, .VBS, ...) and erases and overwrites them with the macro virus code.

2. Downloads another virus and runs it (this virus, called `WIN-BUGSFIX.EXE`, is a password stealing program that sends any passwords it finds in the computer to an email account – mailme@super.net.ph).

3. Sends out virus-infected emails to everyone in the user's address book.
4. Sends the virus over the Internet Relay Chat (IRC) network if the user has mIRC installed.

7.6.2 Helper Applications And Plug-ins

The Netscape browser can only display a few types of files: HTML files (.html), pictures (.gif and .jpg), and plain text. When the browser encounters a sound, image, or video file, it hands off the data to other programs, called **helper applications**, to run or display the file. A helper application is then a special program that is run automatically by the browser when a specific data type is downloaded. We mean with specific data type all data types other than ASCII, HTML, GIF, or JPEG.

Helper applications

Netscape Communications developed a system similar to the helper applications system called **plug-ins**. A plug-in is a module that is loaded directly into the address space of the corresponding browser and is automatically executed when documents of a particular data type (such as a sound) are downloaded. Most popular helper applications have been rewritten as plug-ins, including the Adobe Acrobat reader to display PDF files, the RealAudio player to play sound files, and the Macromedia Shockwave player to play animated video sequences.

Plug-ins

From the security point of view, helper applications and plug-ins can be dangerous. Helper applications and plug-ins run on the user's computer system taking their input from arbitrary (trusted and untrusted) remote computer systems that can be malicious. A user that will download a helper application or plug-in is generally not sure that he/she is downloading an authentic copy. In this case, the user can download a modified version (of the helper application or plug-in) that can be used to abuse his/her computer system.

An interpreter for a general purpose programming language is a powerful helper application. Given the appropriate input, it can open, read, modify, or delete any file on a computer system. Note that many programming languages such as Java allow programs to open network connections and look for security weakness on other computers. For this reason, interpreters for a general purpose programming language are not to be used or configured as helper applications. Other programs should not be used or configured as helper applications [Opp00]:

1. Any programs that includes Microsoft's Visual Basic scripting languages (such as Microsoft World and Excel unless the macros feature is turned off);
2. The scripting languages Perl, Python, and Tcl/Tk;
3. UNIX shells, such sh, csh, tcsh, or any other UNIX shell;
4. The DOS command shell `COMMAND.COM`;

51 See http://www.unifiedchaos.com/.

5. Any Postscript interpreter other than GhostView.

7.6.3 Scripting Languages

Scripting languages

In the context of the Web, **scripting languages** are an intermediate stage between HTML and programming languages such as Java, C++, and Visual Basic. HTML is generally used for formatting and linking text. Programming languages are generally used for giving a series of complex instructions to computers. The scripting languages fall somewhere there between. They are easy to learn, easy to use and allow users to automate some aspects of system operation. In this section, we deal with two popular scripting languages: JavaScript and VBscript.

JavaScript

JavaScript is a scripting language that Netscape developed to make animation and other forms of interaction more convenient. JavaScript codes are contained within the HTML code of the Web page and are interpreted by the browser as it reads them in contrast to Java programs that are downloaded separately from the HTML page. JavaScript code is usually inserted between tags: `<SCRIPT>` and `</SCRIPT>` (so that they will be recognized by JavaScript-enabled browsers). The language within the `<SCRIPT>` tag can be specified as `<SCRIPT language=JavaScript>`.

JavaScript allows HTML files to command the browser. So JavaScript code can create new windows, fill out fields in forms, jump to a new URL, process image maps locally, change the content of an HTML file, compute mathematical results, and/or perform other functions.

From the security perspective, JavaScript code is more secure than code written in Java or in any other programming language. That is because JavaScript does not contain methods that enable either the direct access to the client's computer file system, or the direct opening of connections in other computers on the network.

Denial of service attack
Privacy violation

JavaScript security problems can be resumed in two main areas: **denial of service attacks** and **privacy violation**.

1. Denial of service attacks
   JavaScript can be used to mount denial of service attacks against the users of Web browsers. These attacks can be resident on Web pages or they can be sent to users with JavaScript-enabled email readers. For example, an attacker can write a JavaScript code that can easily overwhelm a computer by simply requiring too much of the CPU or memory resources. This can occur, if the written JavaScript code tries to solve a computationally expensive problem. Another kind of denial of service attacks is to write a JavaScript code that can open a large number of windows on the victim's screen.

2. Privacy violation
   Since JavaScript code runs inside the browser itself, it potentially has access to any information that the browser has. This fact can be used to mount several

attacks against the user of the browser. For example, JavaScript could be used to create forms that automatically fill themselves with private data and send these data to a remote attacker's email. This feature can be applied by an attacker who will collect email addresses of people visiting his Web page. Moreover, JavaScript code has access to the user's browser history mechanism. This allows an attacker to discover the URLs of theWeb pages that a victim had visited during his HTTP session. Finally, a JavaScript program running on one window could monitor the URLs of Web pages visited in other windows. Fortunately, most of these problems have been corrected in newer versions of Netscape Navigator and Microsoft Internet Explorer. Of course, this does not mean that all possible problems are discovered and thereafter solved. For example, an exploit that could potentially affect Netscape 6 mail users is discovered[52]. This exploit could allow the originator of an email message to include hidden JavaScript code in an attachment so that the originator is copied on all forwarded versions of the message [N01].

Spoofing attack

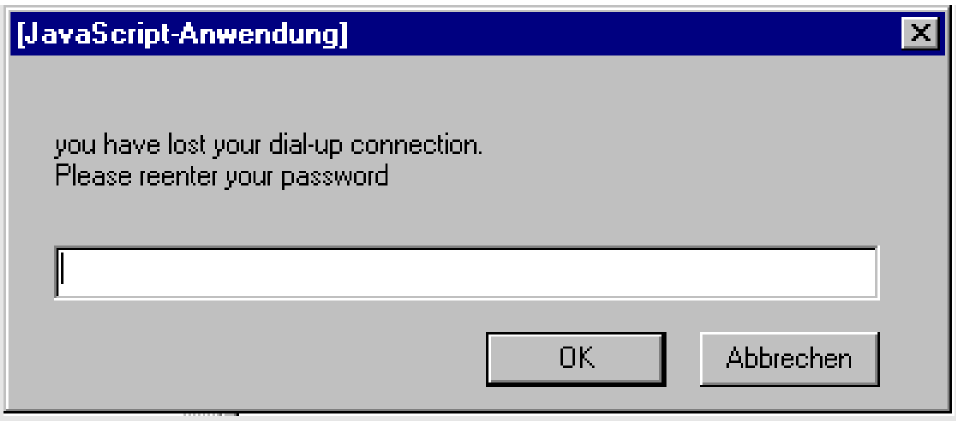
JavaScript enables **spoofing attacks**, too. For example, it can be used to spoof username and/or password of the user by means of creating username/password panel.

Example 7.6-2: Password spoofing
Using the following JavaScript code:

```
<SCRIPT language=JavaScript>
password = prompt("you have lost your dial-up connection.
Please reenter your password","");
</SCRIPT>
```

the following window (see Fig. 7.6-1) can be generated on the browser side.

52 At this writing, Netscape 6 is a relatively new version.



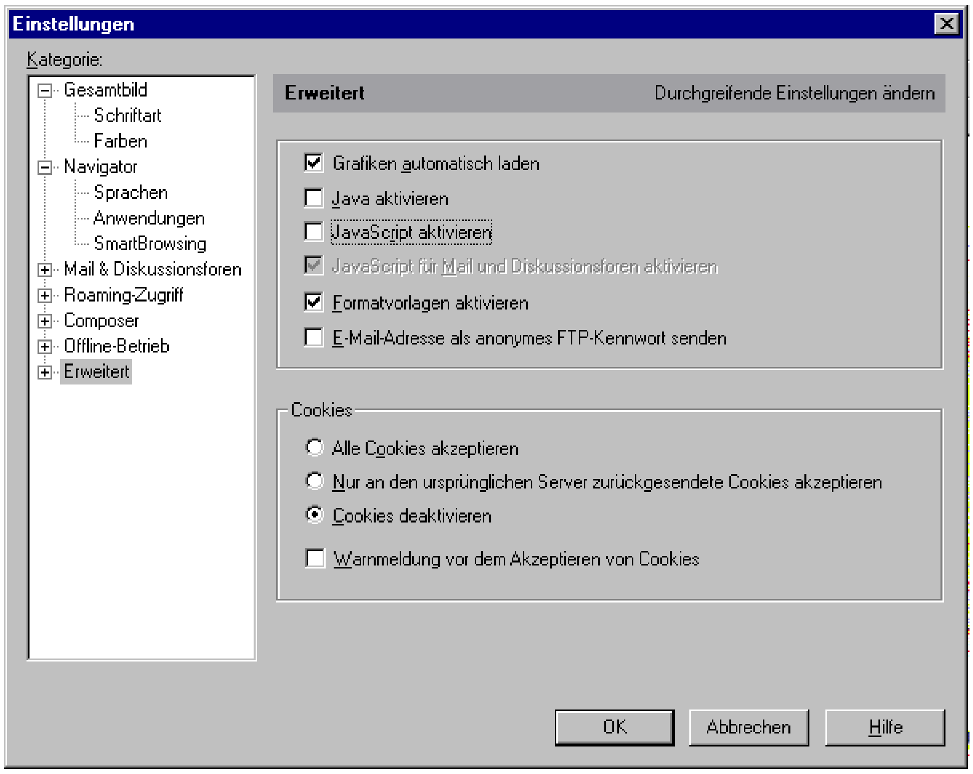
[JavaScript-Anwendung]

you have lost your dial-up connection.
Please reenter your password

Fig. 7.6-1: Password attack.

A careless user can fill out the field and reveal his password. It is possible to enhance the JavaScript code (given above) so that the password – typed by a careless user – can be sent to a remote email account.

In fact, exploiting the JavaScript features, other attacks could be probably discovered and mounted against the user of the browser. An effective protection against all possible attacks is to disable JavaScript in the browser (see Fig. 7.6-2)



Einstellungen

Kategorie:

- Gesamtbild
  - Schriftart
  - Farben
- Navigator
  - Sprachen
  - Anwendungen
  - SmartBrowsing
- Mail & Diskussionsforen
- Roaming-Zugriff
- Composer
- Offline-Betrieb
- Erweitert

Erweitert Durchgreifende Einstellungen ändern

- [✓] Grafiken automatisch laden
- [ ] Java aktivieren
- [ ] JavaScript aktivieren
- [✓] JavaScript für Mail und Diskussionsforen aktivieren
- [✓] Formatvorlagen aktivieren
- [ ] E-Mail-Adresse als anonymes FTP-Kennwort senden

Cookies

- ○ Alle Cookies akzeptieren
- ○ Nur an den ursprünglichen Server zurückgesendete Cookies akzeptieren
- ● Cookies deaktivieren
- [ ] Warnmeldung vor dem Akzeptieren von Cookies

Fig. 7.6-2: Disabling JavaScript/Java by the browser.

VBscript

Unlike the JavaScript language which is available for both Netscape Navigator and Microsoft Internet Explorer, VBscript runs only on Microsoft Internet Explorer. Similar to JavaScript, VBscript can be used in denial of service and privacy violation attacks as well as in other attacks.

7.6.4 Java Applets

Java is an object-oriented, general-purpose programming language that has syntax similar to C++. Although Java was thought of as a language for the World Wide Web, Java is in fact a general-purpose computer language that can be used for writing any code from simple programs to complicated applications.

What initially distinguished the typical Java implementation from other computer languages is the run-time environment. Instead of being compiled for a particular microprocessor, Java programs are compiled into a processor-independent bytecode. This **bytecode** is loaded into a computer's memory by the **Java Class Loader**. Finally, the bytecode is executed on a **Java Virtual Machine** (JVM).

Bytecode
Java Class Loader
JVM

The Java Virtual Machine can run Java programs directly on an operating system such as Windows or MacOS; alternatively, the JVM can be embedded inside a Web browser, allowing programs to be executed as they are downloaded from the World Wide Web. Alternatively, it can use a ”just-in-time” compiler to convert the bytecode into the native machine code of the particular computer on which it is running.

Java can also be compiled directly into machine code and run on a target system. Used this way, Java loses its run-time advantage of being able to run on any computer and any operating system that has a JVM, but it retains its advantage of generating code that has automatic memory management [GS95].

Java was not designed to be a secure programming language. But when Java was deployed in the Web, security immediately became a concern. Note that Java has many features making it a safe programming language. For example, the Java language does not possess pointers as in other computers languages. But safety is not security. In fact, Java uses some techniques to limit what a downloaded Java applet can do. The main ones are **Java Sandbox**, the **SecurityManager** class, the **Bytecode Verifier**, and the Java Class Loader as briefly discussed in the following.

Java Sandbox
SecurityManager
Bytecode Verifier

Sandbox

Java programs run on a virtual computer (JVM) inside a restricted virtual space (runtime environment). They are not allowed to directly manipulate a computer's hardware or to make direct calls to the computer's operating system. Sun Microsystems termed this approach (to security) ”the sandbox”.

SecurityManager class

A series of special Java classes allow programs running inside the sandbox to communicate with the outside world. For example, the Java class `FileOutputstream` allows a Java program to open a file for writing to the user's harddisk. To limit danger which can be made by these classes, the developers of Java created a special class, called `SecurityManager`, which is made to be called before any unsafe operation is executed. The `SecurityManager` class determines whether the operation should be allowed or not.

Class Loader

One dangerous attack is to have a malicious program disable the standard `SecurityManager` class or replace it with a more permissive version. Such an attack could be realized by downloading a piece of machine code or a Java applet that exploits a bug in the Java runtime system. To prevent this attack, the Class Loader examines classes to make sure that they do not violate the runtime system.

Bytecode Verifier

The Bytecode Verifier assures that the bytecode that is downloaded could only have been created by compiling a valid Java program.

Java's security policy

Generally, the Java sandbox, SecurityManager class, Class Loader, and Bytecode Verifier collectively implement a Java security policy. But **Java's security policy** is complicated by the fact that the Java language was designed for two fundamentally different purposes. On one hand, Java is a general purpose language for creating application software. On the other hand, Java is a programming language for creating applets that perform animations, create interactive chat systems, etc.

Dual nature of the Java language

These different purposes require different security policies. The **dual nature of the Java language** leads to a much more complicated security model, which in turn leads to more difficulty in enforcement. Java's original implementors envisioned three different security policies that could be enforced by Web browsers that implemented the Java programming language [GS95]:

1. Do not run Java programs.
2. Run Java programs with different privileges depending on the source of the program. Programs downloaded from Web pages would run with severe restrictions. Programs loaded off the user's harddrive would have no restrictions.
3. No restrictions on Java programs. In this case, Java programs can do anything with the computer harddisk, network connectivity, and anything else.

These Java policies are only examples. Other Java policies can be defined and implemented. For example, we give next some policies that could be set for network connectivity [GS95]:

1. No network connectivity: A Java applet can not access the network.
2. Limited network connectivity: A Java applet can only open network connections to the host from which it was downloaded.
3. Limited network connectivity: A Java applet can only open network connections to a host whose name appears in a set of preapproved hosts.
4. Limited network connectivity: A Java applet can only open network connections on a specified port or ports.
5. No restrictions for applets downloaded from particular machines: A corporation might want applets to be downloaded only from the company's internal "intranet" server.
6. No restrictions for "signed" applets: Java applets that are digitally signed by approved secret key have full access to the computer's resources, unsigned applets are restricted.
7. Unlimited connectivity.

The two Web browsers Microsoft Internet Explorer and Netscape Navigator can be configured in such a manner that Java applets can be controlled and monitored by the user. This can be done, for example, through the use of digitally signed Java classes. The higher level of security is reached by disabled the execution of Java programs by the browser (see Fig. 7.6-2).

7.6.5 ActiveX Controls

ActiveX is a collection of technologies, protocols, and APIs developed by Microsoft that are used for downloading executable code over the Internet. The code is bundled into a single file called ActiveX control.

ActiveX controls are plug-ins that are automatically downloaded and installed, when they are needed. Once these plug-ins are not required, they are automatically deleted.

Fundamentally, two kinds of ActiveX controls are to be distinguished [GS95]:

1. ActiveX controls that contain native machine code. These controls are written in languages such as C, C++. The control's source code is compiled into an executable code that is downloaded by the Web browser and executed on the client machine.
2. ActiveX controls that contain Java bytecode. These controls are written in Java or another language that can be compiled into Java bytecode. These controls are downloaded by the Web browser and executed on a virtual machine.

Authenticode technology

Microsoft's **Authenticode technology** is a technology that is used to discover the author of a particular piece of code and determine that the program has not been altered before a download takes place. Authenticode is based on **digital signatures**

Digital signature

Public key infrastructure

and a **public key infrastructure**. Depending on the two kinds of ActiveX controls, Authenticode can be used for different purposes:

1. For ActiveX controls that contain native machine code:
   Authenticode can be used to enforce a simple decision: either download the control or do not download the control. These Authenticode signatures are only verified when a control is downloaded from the Internet.

2. For ActiveX controls that contain Java bytecode:
   Authenticode can be used to enforce a simple decision: either download the control or do not download the control. Authenticode signatures can also be used to determine what access permissions are given to the Java bytecode when it is running.

If an ActiveX control contain machine code and Java bytecode, or if both machine code and Java bytecode controls reside on the same Web site, the capabilities-controlled access provided by the Java system become irrelevant. Unfortunately, security through code-signing has many problems[53]. For example, buggy code, even when it is properly signed, can damage the user's computer.

Exercise 7.6-1:

1. *Give examples of executable content which can affect the network security!*
2. *How can JavaScript be used to launch a denial of service attack against a host?*
3. *Describe shortly how can the risk related with download of Java applets be limited!*

53 For more details, see [GS95].

8 Security in Wireless and Mobile Networks

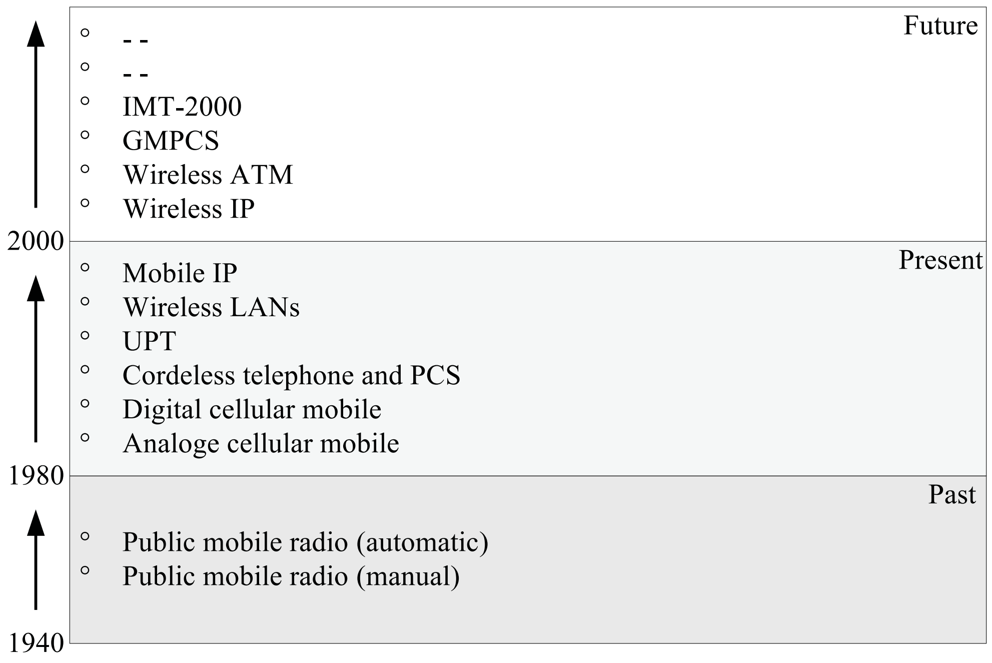
Mobile networks have become a very attractive channel for the provision of electronic services: They are available almost anytime and anywhere. Furthermore, the user acceptance of mobile devices is high. As a result there is a strongly increasing amount of services offered through mobile networks, ranging from simple speech and information services to sensitive applications like banking or electronic commerce. But not only for this reason the security of the data and signalling information is very important. Due to the very sensitive radio path of the data, the communication traffic can be eavesdropped by everyone with very simple devices. After a short introduction to the development of mobile networks, we focus on security in GSM networks. After this, an overwiev of the security mechanisms in UMTS networks, in the WAP protocol, and in the Bluetooth standard is given.

8.1 Development of Mobile Networks

The underlying vision for the emerging mobile and personal communication services and systems is to enable communication with any person at any time, any place, and in any form. Beside providing unlimited reachability and accessability, this vision for personal communications also underlines the increasing need for users of communication services to be able to manage their individual calls and services according to their real-time needs.

The beginning of the radio communication

Radio communication can trace back its origin to the discovery of electromagnetic waves by Hertz in 1888 and the subsequent demonstration of transatlantic radio telegraphy by Marconi in 1901. Mobile radio systems using simplex channels were introduced in the 1920s for police and emergency services. The first public mobile system in the USA was introduced in 1946 and can be considered to be the beginning of the era of public mobile communication services. As illustrated in Fig. 8.1-1, the evolution of public mobile and personal communication services may be divided into three broad periods. The development of the `cellular concept` in the 1970s was a defining event which has played a significant part in the evolution of mobile communication systems and networks around the world.



GMPCS	- Global Mobile Personal Communications by Satellites
IMT-2000	- International Mobile Telecommunications - 2000
PCS	- Personal Digital Cellular
UPT	- Universal Personal Telecommunication

Fig. 8.1-1: Evolution of mobile communication services

The past

At the beginning, mobile radio systems were based on the `trunking principle`. That means that the available frequency spectrum (in the 150 or 450 MHz band) was divided into a suitable number of frequency channnels and a centralized, high power antenna was used to transmit signals to mobile receivers. Large mobile receivers were installed in automobiles, and the telephone sets were also very large. A call originating or terminating on a mobile terminal had to compete for one of the limited number of channels.

The present

Since the initial commercial introduction of `Advanced Mobile Phone system` (`AMPS`) service in 1983, mobile communications has seen an explosive growth worldwide. Beside the frequency reuse capabilities provided by the cellular operation, advances in technologies for wireless access, digital signal processing, integrated circuits, and increased battery life have contributed to exponential growth in mobile and personal communication services. Systems are evolving to address a range of applications and markets, which include digital cellular, cordless telephony, satellite mobile, paging, and specialized mobile radio systems. Data capabilities of these systems are also comming into focus with the increasing user requirements for mobile data communication, driven by the need of mobile Internet access. Whereas analog cellular mobile systems fall in the category of **first generation** mobile systems, digital cellular, low power wireless and personal communication systems are perceived as **second generation** mobile systems.

First generation
Second generation

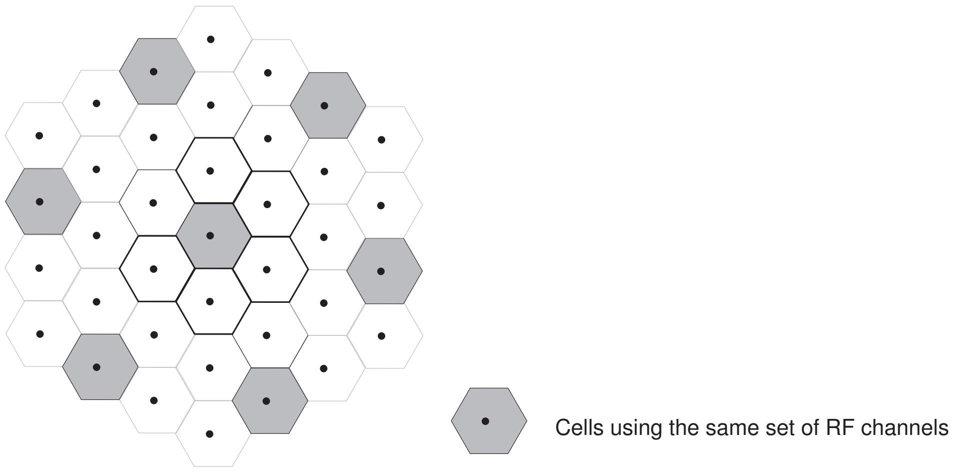


Fig. 8.1-2: The principe of cellular systems

Cellular concept

The **cellular concept** was developed and introduced by the Bell Laboratories in the early 1970s. The principle is to devide a large geographic service area into cells with diameters ranging from from 2 to 50 km/s; for each of them a number of `Radio Frequency` (`RF`) channels is allocated. Transmitters in each adjacent cell operate at different frequencies to avoid interference. Since transmission power and antenna heights in each cell are relatively low, cells that are sufficiently far apart can reuse the same set of frequencies without causing co-channel interference. The theoretical coverage range and capacity of a cellular system are therefore unlimited. As the demand for cellular mobile services grows, additional cells can be added, and as traffic demand grows in a given area, cells can be split to accomodate the additional traffic. Fig. 8.1-2 illustrates an idealized view of a cellular mobile system, where cells are depicted as perfect hexagons. A cellular system should provide the capability to hand over calls in progress, when a mobile user moves between cells. As far as possible the hand over should be transparent to the user in terms of interuption and/or call failure. Hand over between channels in the same cell may also be required for other reasons, such as load balancing, emergency call handling or meeting requirements of transmission quality.

The first digital cellular system specification was released in 1990 by the `European Telecommunications Standards Institute` (`ETSI`) for the `Global System for Mobile Communication` (`GSM`). The GSM, DCS 1800 (`Digital Cellular System`, 1800 MHz version of GSM) and DECT (Digital Enhanced Cordless Telecommunications) systems developed by ETSI form the basis for mobile and personal communication services not only in Europe but in many other parts of the world including North America. The number of GSM subscribers worldwide exceeds 550 million and is still growing rapidly. In the United States, digital cellular standards developed by the `Telecommunications Industry Association` (`TIA`) have been implemented. A third digital cellular system called the `Personal Digital Cellular` (`PDC`) was developed in Japan and is in full commercial operation in that country. The specification for these cellular systems were developed to meet the business and regulatory requirements in specific countries and regions and thus leading to incompatible systems unable to provide global mobility.

The future

While the initial focus of the current generation of mobile and personal communication systems has been circuit switched voice and low bit rate data services, the demand for wide-area as well as local-area wireless data services is rapidly increasing. The emerging industry view is that the main drivers for the next-generation wireless networks will be Internet and multimedia services. Evolution towards high bit rate packet mode capabilities is therefore a key requirement for present and future mobile and personal communication systems. Future mobile systems will represent evolution and enhancements of the present systems in many directions and on many fronts. These directions include increased capacity and coverage, global roaming and service delivery, interoperability between different radio environments, support of high bit rate data, the Internet, multimedia services, global coverage using satellite constellations, and others.

Satellite systems

To complement the cellular and personal communication networks, whose radio coverage is confined to populated areas of the world (less than 15% of the earth surface), a number of global mobile **satellite systems** are in advanced stages of planning and implementation. These systems are generally referred to as `Global Mobile Personal Communications by Satellites` (`GMPCS`). GMPCS systems like Iridium, Globstar and ICO use constellations of `low earth orbit` (`LEO`) or `medium earth orbit` (`MEO`) satellites and operate as overlay networks for existing cellular and PCS networks. They extend the coverage of cellular and PCS networks to any and all locations on the earth's surface.

Third generation

IMT-2000 (International Mobile Telecommunications - 2000) is the standard being developed by the ITU[54] to set the stage for the **third generation** of mobile communication systems. The standard consolidates different wireless environments (cellular mobile, cordless telephony and satellite mobile services). It also ensures global mobility in terms of global seamless roaming and delivery of services. The `Universal Mobile Telecommunication System` (`UMTS`), developed by ETSI, also belongs to the IMT-2000 system.

8.2 Security Mechanisms in GSM Networks

`Global System for Mobile Communications` (`GSM`) is a standard for digital mobile telephony, defined by the `European Telecommunications Standards Institute` (`ETSI`). The first GSM services were started around 1992. Today, this standard is used globally in more than 400 networks operating in more than 170 countries worldwide[55]. The number of GSM subscribers currently exceeds 550 million[56].

54 International Telecommunication Union

55 End of June 2001

56 For more statistical and up to date information about GSM networks see http://www.gsmworld.com/about/membership/mem_stats.shtml.

Aims of the Groupe Special Mobile

The original **aims of the Groupe Special Mobile** which was an initiative of the `European Conference of Posts and Telecommunications Administrations` (`CEPT`) were to provide a harmonized mobile service across Europe, opening up large markets and thus reducing infrastructure and handset costs. In the 1980s the markets within Europe were fragmented between the UK `Total Access Communication System` (`TACS`), the French `Radiocom 2000 System`, the German `C-Netz System`, and the `Nordic Mobile Telephone` (`NMT`) system. What everyone wanted was a new system which could be universally adopted making a single large market for handsets, base sites and switches. The work on the standard was started in 1982, and the first full set of specifications became available in 1990. The responsibility for **GSM standardisation** now resides with the `Groupe Special Mobile` (`GSM`) under the `European Telecommunication Standards Institute` (`ETSI`). An 1800 MHz version of GSM, known as DCS 1800, has also been standardised in Europe.

GSM standardisation

GSM networks in Germany

In Germany the following GSM networks are on air:

- Since July 1992: D1 (DeTeMobil, Deutsche Telekom Mobile Network GmbH) operating at 900/1800 MHz.
- Since June 1992: D2 (Mannesman Mobilfunk GmbH, now Vodafone) operating at 900/1800 MHz.
- Since May 1994: E-Plus (E-Plus Mobilfunk) operating at 1800 MHz.
- Since October 1998: O_2 (formerly known as VIAG Interkom) operating at 1800 MHz.

Before we discuss the security mechanisms in GSM networks, we first overview their features and describe briefly how they work[57].

8.2.1 Brief Overview of GSM Networks

Services in GSM

The most important service supported by GSM is telephony. Other services derived from telephony included in the GSM specification are emergency calling and voice messaging. Bearer services supported in GSM include various asynchronous and synchronous data services for information transfer between GSM and other networks at rates from 300 to 9600 bit/s. This also includes fax and `Short Message Service` (`SMS`). GSM also supports ISDN[58]-like supplementary services: call forwarding, call barring, call waiting, call hold, tele-conferencing, closed user group service and others. Summarized, the **main features of** the initial **GSM** standard include the following:

Main features of GSM

- Fully digital system utilizing the 900 MHz frequency band,

57 Basic knowledge about the principles of mobile networks is expected from the reader. In the brief description below we do not claim completeness.

58 ISDN: Integrated Services Digital Network

- TDMA[59] over radio carriers (200 kHz carrier spacing),
- 8 full-rate or 16 half-rate TDMA channels per carrier,
- User and terminal authentication for fraud control,
- Encryption of speech/data transmission and control data over the radio path,
- Full international roaming capability,
- Low speed data services (up to 9,6 kbit/s),
- Compatibility with ISDN for supplementary services,
- Support of `Short Message Service` (`SMS`).

Reference Architecture and Function Partitioning

GSM system

As shown in Fig. 8.2-1, the **GSM system** comprises of `Base Transciever Stations` (`BTS`), `Base Station Controllers` (`BSC`), `Mobile Switching Centers` (`MSC`), and a set of registers (databases) to assist the mobility management and security functions. All signalling between the MSC and the various registers (databases) as well as between the MSCs takes place using the `Signalling System No. 7` (`SS7`) network, with the application level messages using the `Mobile Application Protocol` (`MAP`) designed specially for GSM. The signalling between the MSC and the national PSTN[60]/ISDN network is based on the national options for SS7, `Telephone User Part` (`TUP`) or `ISDN User Part` (`ISUP`). In order to understand how a GSM network works, we now briefly discuss its most important constituents.

Mobile Station

Mobile Station (MS)

The GSM mobile stations (also called handsets or mobile phones) are portable radiotelephony units that can be used on any GSM system. Power levels supported by the GSM mobile stations currently range from 0.8W to 8.0W, and power saving techniques are used on the air interface to extend battery life. At the time of manufacturing, an International Mobile Equipment Identity (IMEI), which is not easily alterable or removable, is programmed into the terminal. To activate and operate a

SIM card

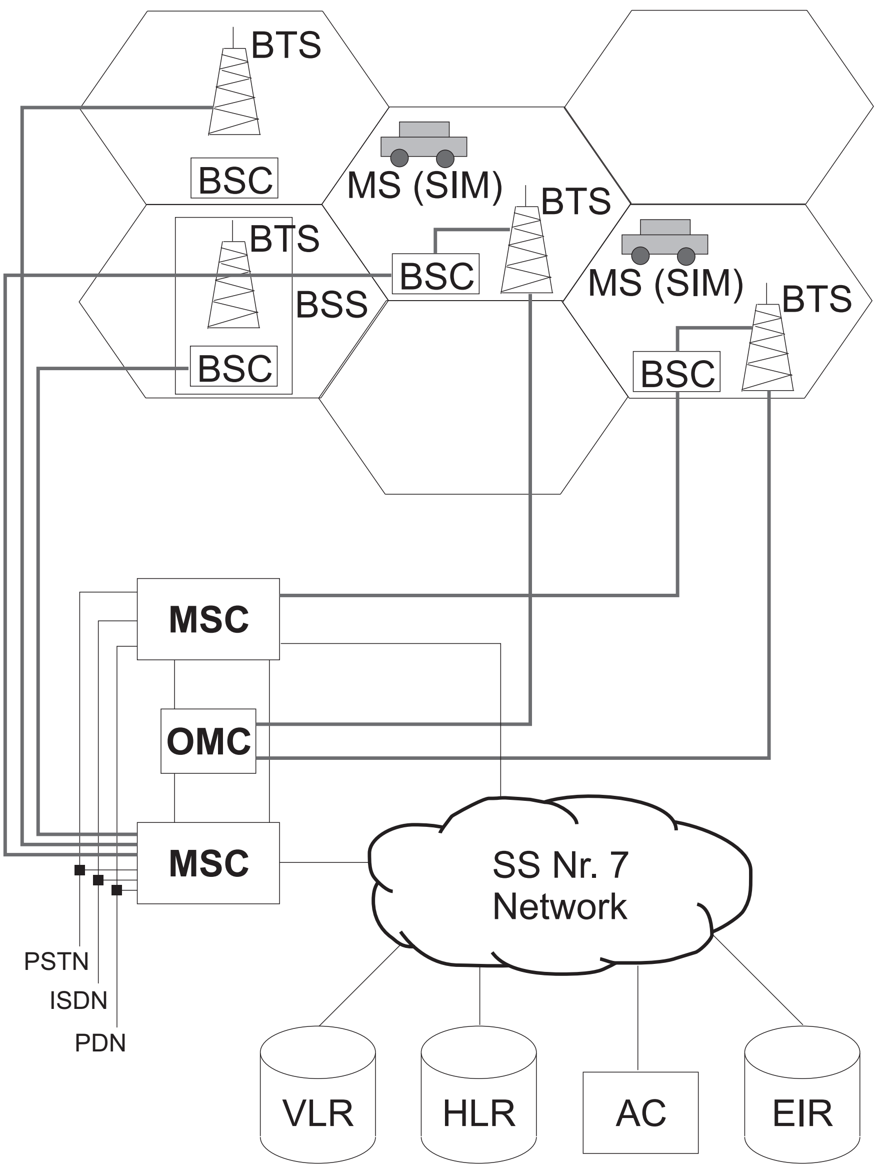
GSM terminal, a Subscriber Identity Module (SIM) is required. The SIM may be contained within the MS, or it may be a removable unit that can be inserted into the

IMSI number

MS by the user. The `International Mobile Subscriber Identity` (`IMSI`) number, which identifies the subscriber uniquely, is programmed into the SIM at the time of service provisioning, along with the appropriate security parameters and algorithms.

59 Time Division Multiple Access

60 PSTN: Public Switching Telephone Network, i.e. the ”classical” analogue telephone network



BSS	-	Base Station System	VLR	-	Visitor Location Register
BSC	-	Base Station Controller	EIR	-	Equipment Identity Register
BTS	-	Base Transceiver Stations	OMC	-	Operation and Maintenance Center
MS	-	Mobile Station	PSTN	-	Public Switched Telephone Network
SIM	-	Subscriber Identity Module	ISDN	-	Integrated Services Digital Network
MSC	-	Mobile Switching Center	PDN	-	Public Data Network
AC	-	Authentication Center	SS7	-	Signalling System 7
HLR	-	Home Location Register			

Fig. 8.2-1: Architecture of a GSM network

Base Station System

Base Station Controller (BSC)

Base Transceiver Stations (BTS)

Base Station System (BSS)

The Base Station System comprises of a **Base Station Controller (BSC)** and one or more subtending **Base Transceiver Stations (BTS)**. The BSS is responsible for all functions related to sending/receiving data and for the radio resource (channel) management. This includes configuration of radio channels, selection, allocation, and deallocation of radio channels, monitoring of radio channel status (busy/idle), encryption of radio interface, assignment of frequency-hop sequence and start time, assignment of effective radiated power (ERP) values to mobile stations, collection of signal quality data from adjacent BSSs, analysis of signal quality data and determination of handoff need, keeping the mobile switching center (MSC) informed regarding hand over activity, and it also includes digital signal processing (transcoding and rate adaption, channel coding and decoding).

Mobile Switching Center

Mobile Switching Center (MSC)

The Mobile Switching Center for GSM can be viewed as a local ISDN switch with additional capabilities to support mobility management functions like terminal registration, location updating, and hand over. Unlike a local switch in a fixed network, the MSC does not administer the mobile subscriber parameters wich are dynamic and are held in the `Visitor Location Register` (`VLR`) associated with the MSC. The MSC performs the following major functions: call setup, supervision and release, call routing, billing information collection, mobility management (registration, location updating, inter-BSS and inter-MSC call hand overs) management of radio resources during a call, echo cancelation, allocate resources to BSS, other MSCs, PSTN/ISDN, and other functions.

Home Location Register

Home Location Register (HLR)

The HLR represents a centralized database that has the permanent datafile about the mobile subscribers in a large service area (generally one per GSM network operator). It is referenced using the SS7 signalling for every incoming call to the GSM network for determining the current location of the subscriber. The HLR is kept updated with the current locations of all its mobile subscribers, including those who may have roamed to another network operator within or outside the country. The routing information is obtained from the serving VLR on a call-by-call basis (for each incoming call the HLR queries the serving VLR). Beside the up-to-date location information for each subscriber (which is dynamic), the HLR maintains the following subscriber data on a permanent basis: `International Mobile Subscriber Identity` (`IMSI`), service subscription information, service restrictions, mobile terminal characteristics, and billing and accounting information (see Fig. 8.2-2).

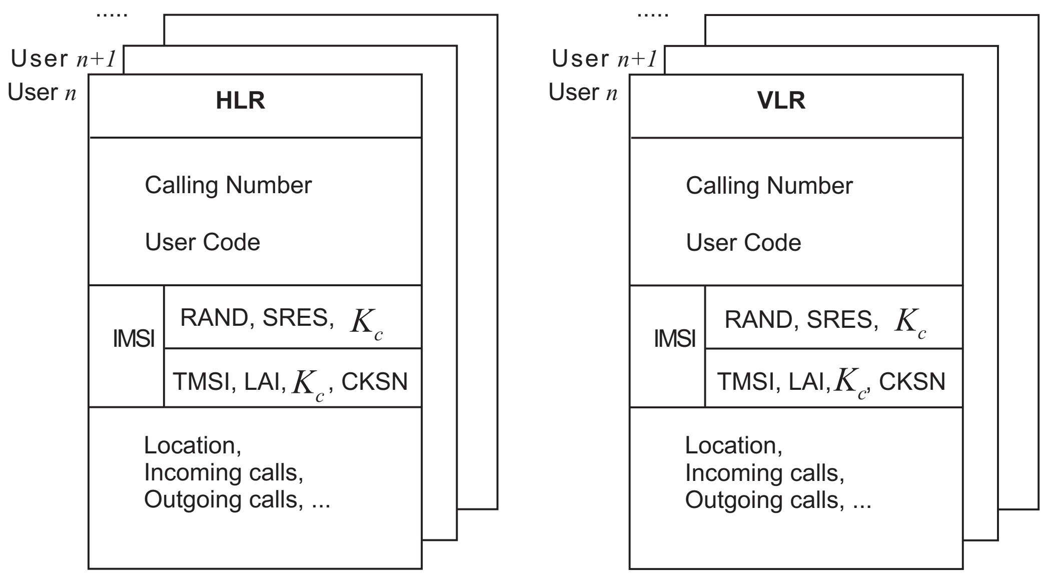


Fig. 8.2-2: Data fields in HLR and VLR

Visitor Location Register (VLR)

Visitor Location Register

The Visitor Location Register (see Fig. 8.2-2) represents a temporary data store and generally there is one VLR per MSC. This register contains information about the mobile subscribers which are currently in the service area covered by the MSC/VLR. The temporary subscriber information resident in a VLR includes the currently activated features, the `Temporary Mobile Station Identity` (`TMSI`), and the current location information about the MS (e.g., location area and cell identities).

Authentication Center (AC)

Authentication Center

The Authentication Center contains authentication parameters that are used on initial location registration, subsequent location updates and on each call setup request from the MS. The AC maintains the authentication keys and algorithms and provides the security triplets ($RAND$, $SRES$, and K_c) to the VLR so that the user authentication and radio channel encryption procedures may be carried out within the visited network. It contains the security modules for the authentication keys and the authentication and cipher key generation algorithms A3 and A8. In the following sections we will discuss the functions of the AC in more detail.

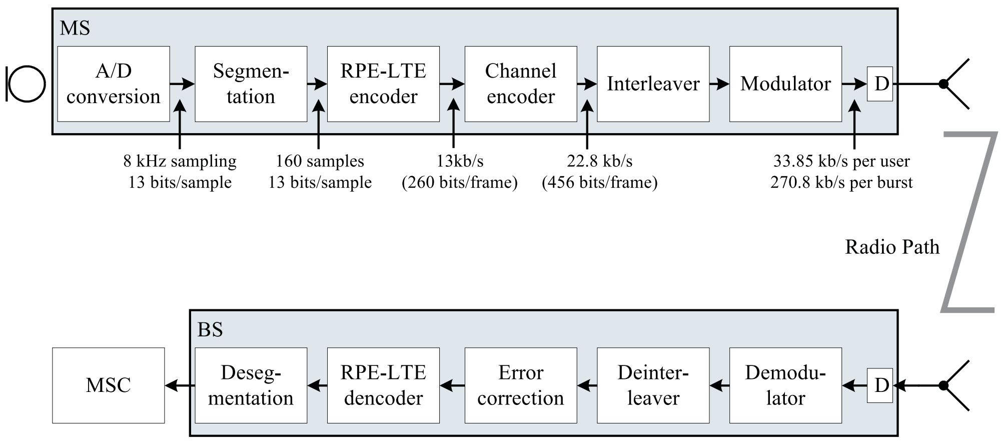
Equipment Identity Register (EIR)

Equipment Identity Register

The Equipment Identity Register maintains information to authenticate terminal equipment to identify and deny service to fraudulent, stolen or non-type approved terminals. The information is in form of white, gray and black lists that may be consulted by the network when it wishes to confirm the authenticity of the terminal requesting service.

Radio Characteristics

In GSM the uplink (mobile to base) frequency band is 890 - 915 MHz and the corresponding downlink (base to mobile) band is 935 - 960 MHz. The GSM uses `Time Division Multiple Access` (`TDMA`) and `Frequency Division Multiple Access` (`FDMA`), where the available 25 MHz spectrum is partitioned into 124 carriers (carrier spacing is 200 KHz). Each carrier in turn is devided into 8 time slots. Each user periodically transmits in every eighth time slot in an uplink radio carrier and receives in a corresponding time slot on the downlink carrier. Thus several conversations can take place simultaneously in the same pair of transmit/receive radio frequencies.



BS	- Base Station
D	- Duplexer
MS	- Mobile Station
MSC	- Mobile Switching Center
RPE-LTE	- Regular Pulse Excitation - Long Term Prediction

Fig. 8.2-3: Speech encoding and modulation in GSM

GSM speech encoding and modulation

Digitalized speech is passed at 64 kbit/s through a speech coder (transcoder), which compresses it to 13 kbit/s. The most important steps of the process of information transfer for speech signals in the uplink direction across the radio interface are illustrated in Fig. 8.2-3 and will not be discussed here in detail. A similar process is applicable for the downlink.

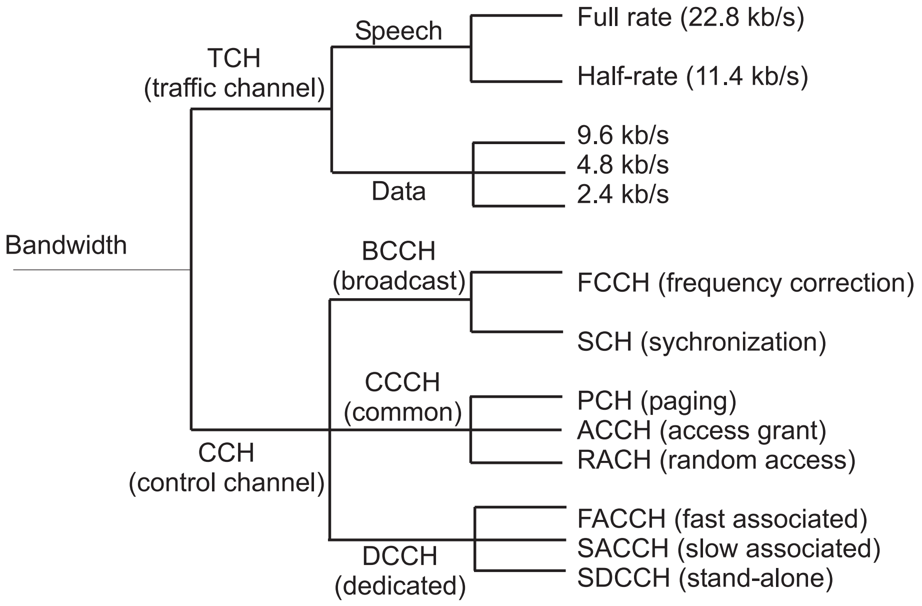


Fig. 8.2-4: Logical channel structure in GSM

Logical channels in GSM

There are two distinct categories of **logical channels in GSM**: `Traffic Channels` (`TCH`) and `Control Channels` (`CCH`). In the initial GSM specification only the full-rate speech and data channels were defined as traffic channels. The three types of control cannels are `Broadcast` (`BCCH`), `Common` (`CCCH`), and `Dedicated` (`DCCH`) channels, each of them is further subdivided. Each type of traffic channel is always associated with a control channel.

To reduce the effects of multipath fading when a mobile station is stationary or moving slowly, GSM provides **frequency hopping** capabilities on the radio interface. Frequency hopping also provides added security against unauthorized eavesdropping on a call in progress. It is achieved in such a way that the MS sends and receives on different frequencies, according to the calculated sequence on both sides of the air interface.

Frequency hopping

8.2.2 Specific Security Threats

A basic design requirement of GSM was the security of communication. Threat analysis and security services are similar in nature for most mobile communication systems. The most obvious of them are:

1. Illegitimate use of a service,
2. building of profiles,
3. loss of confidentiality (eavesdropping the connection), and
4. loss of data integrity on the air interface.

Illegitimate use of service

The **illegitimate use of service** is probably the main security threat in a GSM network. It is clearly important that billing should always be possible and only the subscriber, who has caused the charge, is billed for it. The purpose of a masquerading attack may, however, not just be to defraud the network by having the charges transferred to `any` subscriber but to impersonate a `specific` subscriber. This could then be used to ”show” that the subscriber made a call from a particular place at a particular point in time. Authentication of the subscriber by cryptographic means is needed. Additionally the use of a security device for storing data and algorithms for the authentication process are essential requirements.

Confidentiality of the user identity

At the set up of a session the identity of the subscriber has to be sent to the network by the MS. Furthermore, due to the constant update of the location of a MS, which is necessary for a proper and timely delivery of mobile calls, the network has to know the location of the subscriber card down to the cell in which the MS is active. Measures need thus be taken to protect the confidentiality of the identity of the subscriber.

Building profiles

The security procedures should also prevent unauthorized parties from **building profiles**, i.e. tracing the identity and location of the subscribers as they roam within or outside the home network. To protect the identity and the location of the subscriber, the appropriate radio signalling (control) channels are enciphered and `temporary identities` (`TMSI`) instead of the actual identity (IMSI) are used over the radio path.

Compromising the SS7 messages

The security of the global Signalling System No. 7 (SS7) network as a transport system for sensitive signalling messages between different telecommunication network elements is open to major compromise. Messages can be eavesdropped, altered, injected or deleted in an uncontrolled manner. This messages are exchanged between network elements which sometimes belong to different network operators. Examples in GSM are particulary sensitive authentication data of mobile subscribers which have to be transported from the Authentication Center (AC) to the Visitor Location Register (VLR) in order to authenticate the subscriber. The authentication data can be compromised either during its transport between the home environment and the serving network, or by unauthorized access to databases. This can lead to serious attacks, such as forcing to use a compromised cipher key[61] or impersonating the user[62].

Loss of confidentiality Listening to the communication on the air interface between a Mobile Station and a Base Station can hardly be prevented. Generally, users of wireless networks are more at risk than users of classical ”wired” networks, because the radio channels can be eavesdropped with very simple devices (digital receivers with features similar to those of base stations or deivces for network testing). The interception of user data or user related signalling information may result in a loss of confidentiality of this data or of the user’s identity with respect to the outside world. Both traffic and signallig channels (for example, the information which two user communicate) can be eavesdropped. One of the novel features of GSM is the enciphering of the radio link (both traffic and signal channels) to protect user and signalling data against eavesdropping. Special ciphers have been developed for this purpose.

Loss of data integrity The loss of data integrity on the air interface is also simpler than in a ”wired” network. Adequate equipped persons can huddle on the role of the receiver. So, the sender can not be sure, that his data reaches the right receiver and that the data he receives has the appropriate source (sender). Moreover, the digital representation of data enables lots of manipulation possibilities of the data (deleting part of the data, inserting new data, changing data), which can not be noticed on the receiver side without adequate mechanisms for protecting the data integrity.

To summarize, we can say that for protection of the network operators and users against attacks, the following security features have to be `mandatorily` implemented:

- User identity authentication,
- signalling information confidentiality,
- user data confidentiality, and
- user data integrity.

Note that the privacy mechanisms in GSM (encryption and use of temporary identities) are used only over the radio path and not within the fixed network infrastruc-

61 The intruder obtains a sample of authentication data and uses it to convince the user that he is connected to a proper serving network, and forces the use of a compromised cipher key. The intruder may force the repeated use of the same authentication data to ensure the same encryption key will be used for many calls. This leads to continous eavesdropping.

62 The intruder obtains a sample of authentication data and uses it to impersonate a user towards the serving network.

ture, where the communication is carried out in clear mode (ie. unencrypted), as it is in PSTN[63] and ISDN.

8.2.3 The Subscriber Identity Module (SIM)

SIM as security device

The **Subscriber Identity Module (SIM) is a security device** integrated in the users mobile station (MS) which contains all necessary information and algorithms to authentificate the subscriber to the network. It also adds a new dimension of mobility to the subscriber as it is a removable module and may be used in any mobile equipment.

The SIM as a smart card which comes in two formats. The ID-1 SIM has a format of a credit card. The Plug-in SIM, which is used mainly with mobile devices too small to support an ID-1 SIM, has a size of 25 by 15 mm. The electrical and mechanical interfaces are in line with the relevant International Standards for Integrated Circuit (IC) cards.

Microprocessor in the SIM

To achieve its main task of authenticating the subscriber to the network, the **SIM contains a microprocessor**, which consists of a CPU and three types of memory. The Read Only Memory (ROM) usually contains the operating system of the card, the code for the GSM application and the security algorithms $A3$ and $A8$[64]. The Random Access Memory (RAM) is used for the execution of the algorithms and as a buffer for the transmission of data. Subscription specific data, such as the key K_i or IMSI, network related information, such as TMSI and LAI, which are needed to be updated by the network, and subscriber related information such as dialing numbers, are stored in non-volatile erasable memory (EEPROM, Electrically Erasable Programmable Read Only Memory).

Access to the SIM

The card's operating system controls access by the outside world to all data stored in the card. User **access to the SIM** is controlled by a `Personal Identification Number` (`PIN`). The PIN can be freely chosen in the range of 4 to 8 digits and is stored in the EEPROM of the chip. The PIN does not leave the chip because the verification of the PIN presented by the user is carried out by the microprocessor of the SIM. To protect the user against trial and error attacks, the microprocessor controls the number of consecutive false PIN entries. After three false entries the card is blocked and will refuse to work. A blocked SIM does not even send its identity in form of IMSI or TMSI as these data fields are protected against reading by the security level PIN. The user can only ”unblock” a SIM by presenting the so-called `PIN Unblocking Key` (`PUK`, another identification number consisting of eight digits) to the card together with a new PIN. The PUK can be stored in the home network and given to the user if the necessity arises. The access of a SIM to a mobile service can easily be suspended by blacklisting the subscription in the HLR/AC.

63 Public Switched Telephone Network

64 The Algorithms $A3$, $A5$, and $A8$ are explained later.

8.2.4 Security Services

Security services

The specification of the security related network functions and the parameters for the cryptographic algorithms needed to provide the **security services** are specified in GSM 03.20 ([GSM0320]) which form the basis for the implementation of these features. From the user's point of view it is not relevant whether the user-related data to be protected is contained in a traffic or a signalling channel. We may therefore say that GSM provides three security services:

- Authentication for the corroboration of the identity of the user,
- enciphering for the confidentiality of user-related data, and
- temporary identities (pseudonymes) for the confidentiality of the user identity.

Authentication of the Subscriber

Purpose of authentication

Authentication is the corroboration that an entity is the one claimed or, in this context, the verification of the identity of the SIM card. The user authenticates itself to the SIM card with its PIN, and the SIM card authenticates to the network with a cryptographic strong authentication algorithm. Subscriber authentication is of major interest to each operator (protect the network against unauthorized use, correct billing, preventing masquerading attacks). The authentication algorithm in GSM is denoted as A3 and is implemented in the Authentication Center (AC) of the home network, i. e. the Home Public Lands Mobile Network (HPLMN), and in the SIM. The method employed between HLR/AC and a SIM is a challenge-Response mechanism[65] using cryptographic secure random numbers. Thereby, a 128-bit authentication key K_i is used, which is kept in the SIM card (subscriber side) and in the AC (network side).

65 More about challenge-and-Response algorithms can be found in [Kaderali00b]

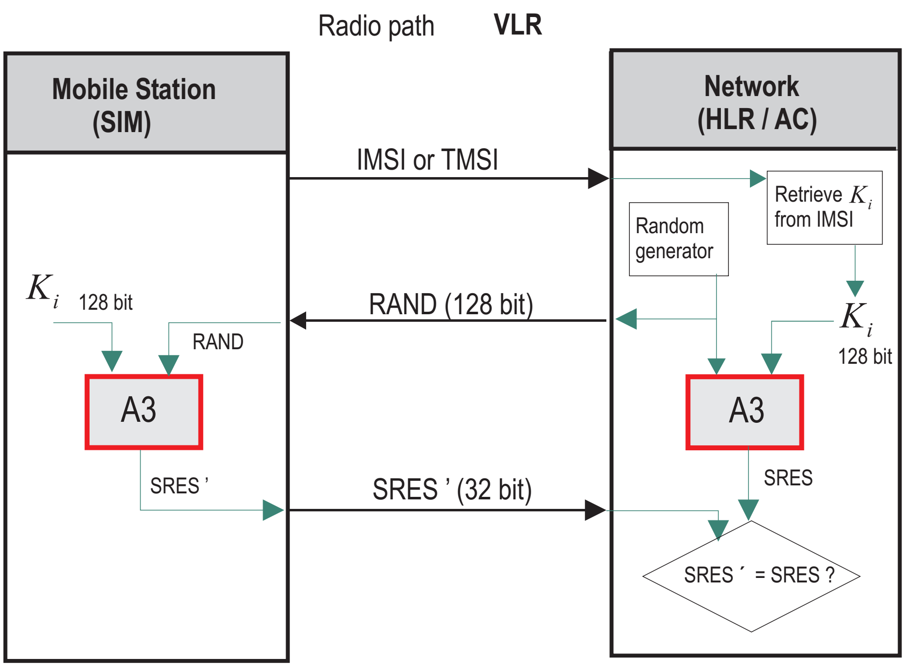


Fig. 8.2-5: Authentication protocol

Authentication protocol

Fig. 8.2-5 shows the basic procedure for the authentication of the SIM by the network. After validating the identity of the SIM, the VLR sends an authentication request to the network. This request contains the IMSI (or TMSI) which is needed to retrive the secret on the network side (which is an individual subscriber authentication key K_i). The network then generates a non-predictable 128-bit random number RAND which is sent to the MS as a challenge (via the VLR). This challenge changes each time the protocol is run. To compute the Signed Response SRES to the challenge RAND, the SIM uses the algorithm A3 with RAND and the key K_i (stored in the SIM) as input data. The algorithm A3 is actually a one-way function with a 32-bit output SRES. SRES′ computed by the the SIM is then transmitted to the VLR (which may be in a foreign network). There it is compared with the value SRES computed by the home network. The AC uses the same RAND and the key K_i which is associated with the identity claimed by the subscriber. The MS is granted access to the network by the VLR only if the value of SRES′ received from the MS equals the value for SRES from the HLR/AC. Only in this case it can be assumed that the SIM is in possession of the right subscriber key K_i and that its identity is the one claimed.

The authentication process takes less then 500 ms. When a user moves to a new VLR, the new VLR will normally establish the subscriber's identity by requesting the IMSI from the old VLR (see section Section 8.2.5).

Note that the individual subscriber keys are not transmitted over the network – they are only used in the challenge-response protocol for authentication and key agreement.

Authentication of the Network

The MS can not authenticate the network

GSM does not provide the means for the SIM (i.e. the mobile station) to authenticate the network. When specifying the security features of GSM this functionality was considered to be of limited use and the idea of administrating data in the SIM over the air interface had not been of widespread interest. This has changed considerably in the last few years, because the availability of the so-called SIM Application Toolkit leads to new posibilities and security problems.

Security problems of SIM Application Toolkit

The **SIM Application Toolkit** offers a platform for new operator specific services such as loading of data into the SIM by use of the short message service. A typical example would be the update of service numbers or the creation of new data fields in the SIM. The SIM executes the command internally, and is dependent on the data to be updated and the commands to be executed; the SIM thus has to be sure that the sender of the short message as well as its contents are genuine and have not been altered in an unauthorized manner. Work has begun on providing a security standard for the short message service as part of the SIM Application Toolkit.

Enciphering

The purpose of this security service is to ensure the privacy of the user information carried in both traffic and signalling channels and of user-related signalling elements on the radio path. The activation of this service is controlled by the network. It is started by the base station by sending a `start_cipher` command to the MS.

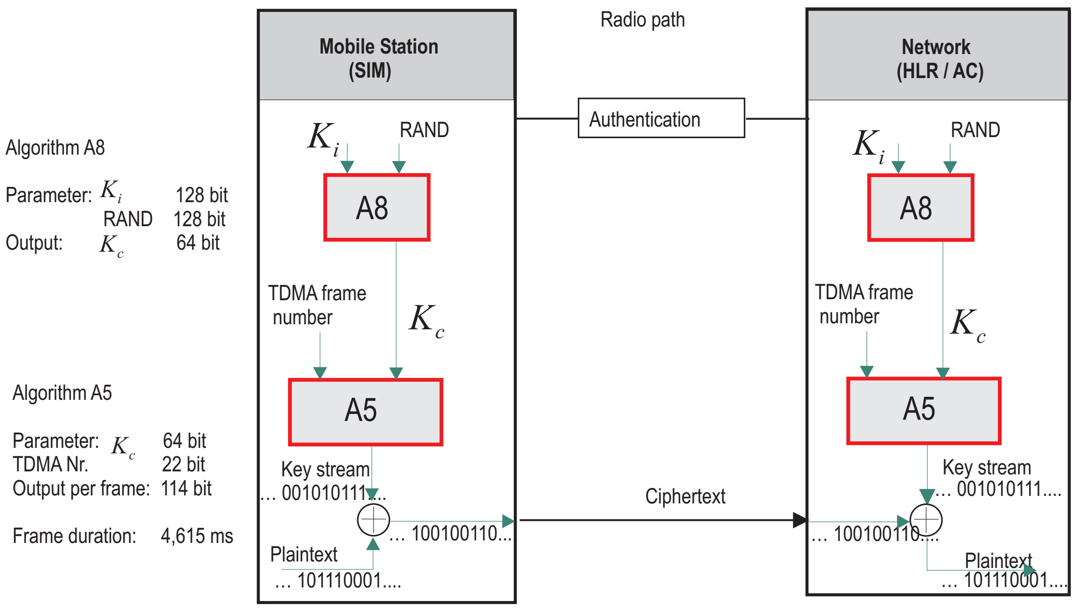


Fig. 8.2-6: Cipher key generation and enciphering

A standard cipher algorithm called A5 is contained as dedicated hardware in mobile equipment and base stations. A5 is a stream cipher[66], and because of its high encryption rate is suitable for real time applications such as telephony.

Key generation and enciphering

The plain text is organised into blocks of 114 bits as this is the amount of data which is transmitted during a time slot. The key stream, which is a sequence of bits to be XORed (modulo 2 addition) with the data block, is produced by the algorithm `A5` as an output block of 114 bits. The generation of the key stream from the `A5` algorithm is controlled by the key K_c (input parameter). This key is derived in the SIM as part of the authentication process using the network operator specific key generation algorithm `A8` and the same `RAND` and K_i as in the authentication algorithm `A3`. The process of cipher key generation and enciphering is shown in Fig. 8.2-6.

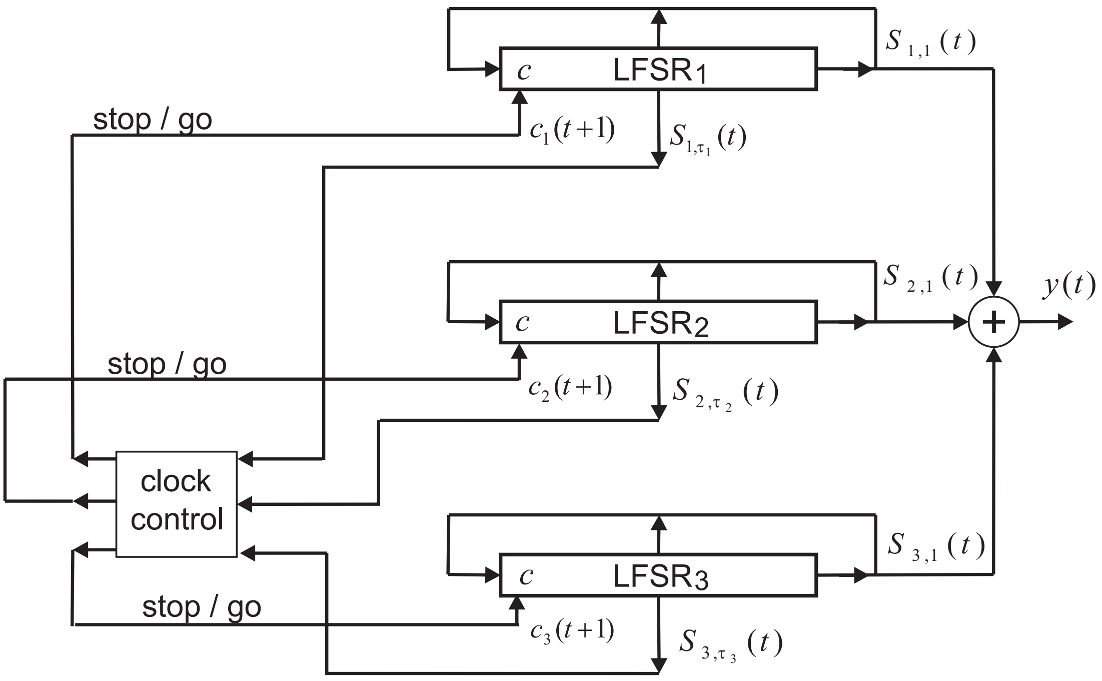


Fig. 8.2-7: `A5` keystream generator

The A5 cipher

A common type of keystream generators for additive stream cipher applications consists of a number of possibly irregularly clocked `linear feedback shift registers` (`LFSRs`) that are combined by a function with or without memory. The binary stream cipher `A5` consists of three short LFSRs (see Fig. 8.2-7) of total length 64, which are mutually clocked in a stop/go manner. The LFSR lengths are 19, 22, and 23 respectively, and the total length is 64. All feedback polynomials are sparse. Middle taps in each of the LFSRs are used to define the clock-control sequence. The clocking rule is such that at least two LFSRs are effectively clocked per each output bit, and the keystream sequence is formed as the bitwise sum of the three stop/go clocked LFSR sequences. The 64-bit secret key is nonlinearly combined with a 22-bit public key (frame number, which is different for every new message) to form the LFSR initial states. The first 100 output bits are discarded and the message length is only 114 bits (frequent resynchronisation). The source code of the software realisation of `A5` is given in [Schneier96].

66 Stream ciphers have been extensively treated in [Kaderali00b]

Weakness of the A5 cipher

Standard cryptographic criteria such as a large period, a high linear complexity, and good statistical properties are fulfilled by the `A5` key stream generator. However, such a generator may in principle be vulnerable to various divide-and-conquer attacks in the known plaintext scenario, where the objective is to reconstruct the secret key controlled initial states from the known keystream sequence. Other attacks on the `A5` algorithm have also been published: the time-memory trade-off attack based on the birthday paradox, the internal state revision attack via branching([Golic97]), or the attack on the tap-structure [Shamir99]. The weakness of `A5` is that its registers are short enough to make this attacks possible in realistic time.

But this weakness was politically intended. Originally, it was thought that GSM's cryptography would prohibit export of the phones to specific countries. Now some officials are discussing whether `A5` might harm export sales, implying that it is so weak as to be an embarrassment. Rumor has it that the various NATO intelligence agencies had a catfight in the mid 1980s over wheather GSM encryption should be strong or weak. The Germans wanted strong cryptography, as protection against the Soviet Union [Schneier96] but were overruled by some other countries. `A5` is a french design.

However, the biggest security hole in the GSM system is that links between the base stations are not encrypted at all. This not only means that all telephone conversations can be easily eavesdropped by the telephone company, but also that the equipment in the fixed part of the GSM network is vulnerable to attacks from outside.

Temporary Identities

Pseudonymes
Temporary identities

Profile building in GSM, i.e. tracing the identity and the location of the subscribers by unauthorized parties, is prevented by employing the techniques of **pseudonymes** or **temporary identities**. Over the radio path, even in the process of authentication, temporary identities (TMSI) are used instead of the actual identities (IMSI).

When a user (subscriber) logs into a network, his identity has to be disclosed to the network. Rather than sending the `International Mobile Subscriber Identity` (`IMSI`) number, which uniquely identifies the subscriber world-wide, a `Temporary Mobile Subscriber Identity` (`TMSI`) is transmitted by the mobile station in the most instances. The purpose of this temporary identity is to undermine the possibility of an intruder to gain the resources used by a subscriber. Furthermore it prevents the tracing of a subscribers location and the matching between the users and the transmitted data.

First authentication

Clearly, the IMSI has to be used for the set up of a session if there are no other means to identify a mobile subscriber. This happens for instance, when the subscriber uses the SIM for the first time or when data is lost in the VLR, where the subscriber is temporarily registered. When the SIM is used for the first time, the MS will read the default Temporary Mobile Subscriber Identity (TMSI) stored in the SIM and send this value to the VLR. As this is a default value, the VLR will request the IMSI from the MS. It then assigns a TMSI to the subscriber and transmits it (after successful authentication and activation of the cipher, see Fig. 8.2-8 a.) in an enciphered form to the MS. The MS deciphers the data and stores the TMSI and information about the present location (`LAI`, `Location Area Identity`) in the SIM.

The mobile station will use this TMSI in the authentication procedure for the next access to the network. So, even if the intruder knows the IMSI of the subscriber, it is not enough to trace his sessions.

Next authentication

At the next log-in the TMSI will be sent in clear mode (ie. unencrypted) to the network, and after the authentication procedure a new TMSI will be assigned by the VLR, and sent to the mobile station as enciphered data (the TMSI, which is sent in clear mode to the network, was sent encrypted to the MS in the last session).

Then a new TMSI will be assigned by the VLR and sent to the MS as enciphered data (Fig. 8.2-8 b.). The mobile station saves the new TMSI and LAI and acknowledges this to the network. The TMSI must be updated whenever the MS enters a new location area and will be changed several times during a session.

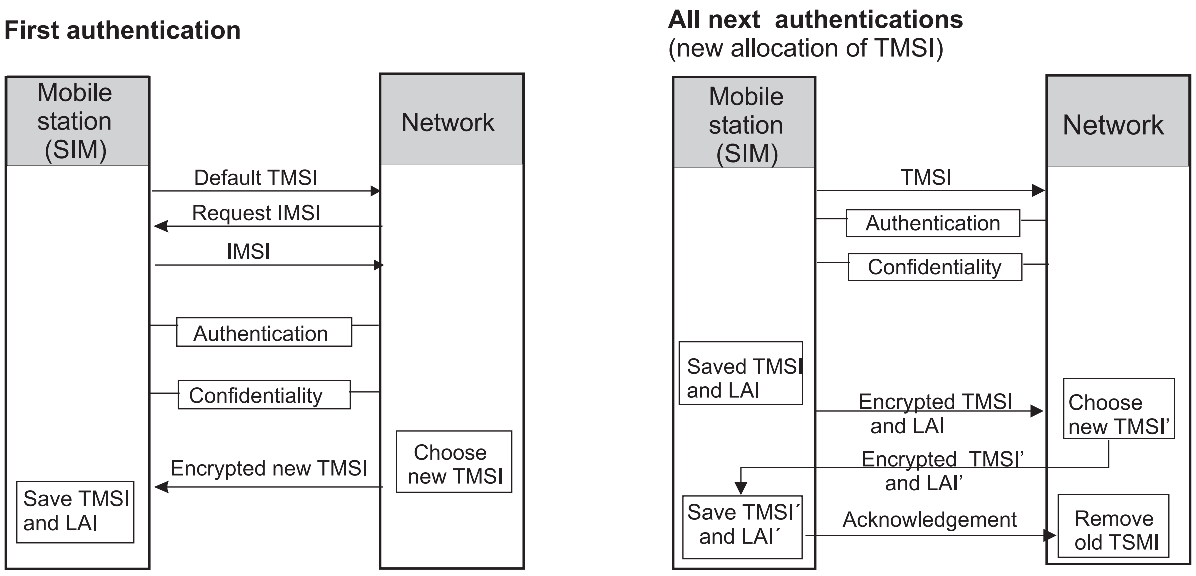


Fig. 8.2-8: Pseudonymes technique with temporary identities

Although the TMSI consists of only five digits, the subscriber is uniquely identifiable. The TMSI is unique within the location area where the MS moves, and the location area identity (LAI) is always used in conjunction with the TMSI. As TMSI, IMSI, and LAI are stored in the VLR, the identity of a subscriber moving to a different VLR can easily be validated by the new VLR. It knows the ”old” VLR from the LAI which the MS had sent together with the TMSI at log-in and it then obtains the subscribers IMSI from the old VLR. Apart from the initial session or a malfunctioning of the system, the IMSI will not be used over the radio path for call set up. The GSM 3.20 standard specifies eight procedures for the allocation of a new TMSI depending on the structure of the network and possible data losses.

8.2.5 Realisation of Security Services

Authentication Center (AC)

Role of the AC

The central points of the security services in GSM networks are the Authentication Centers (AC), which are assigned to each Mobile Switching Center (MSC). They generate and store the IMSI and the authentication key K_i for each subscriber, which are also stored in the subscriber's SIM.

Using the algorithms A3 and A8, a triplet of random bit streams RAND, signed responses SRES and session keys K_c are generated in the AC in advance, and stored in the associated HLR, in conjunction with the appropriate identifier (see Fig. 8.2-9).

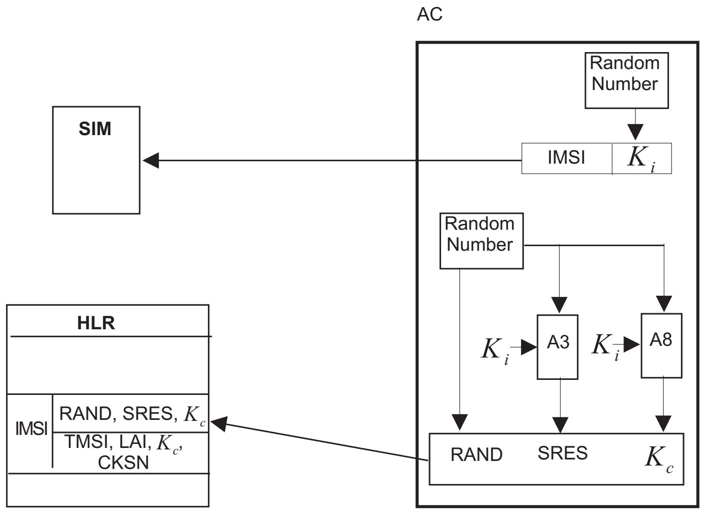


Fig. 8.2-9: Authentication center

The specific security and administrative requirements for an authentication center are not standardised but left to each network operator. The GSM 3.20 standard only states that the individual subscriber authentication keys K_i are stored in an AC and that an AC also contains the authentication algorithm A3 and the cipher key generating algorithm A8.

A malfunction or a temporary loss of the information stored in an AC would have severe consequences for the security as it affects the generation of the authentication triplets. Since other information about the subscriptions, including the possible black lists of barred subscriptions, is contained in HLR, it is logical to "integrate" the AC into the HLR. In networks with more than one HLR, the backup and overload facilities can be distributed over several HLR /ACs.

Key management

Key management is a major issue when designing an AC. The method used for generating and storing potentially several million individual subscriber authentication keys and the handling of the authentication requests are of importance for both the secure and the smooth running of the network.

There are two standard methods to generate keys. They may be generated by using a random number generator or by deriving them from user related data with the help of an algorithm under the control of a `Master Key` (`MK`). Both methods have their advantages and disadvantages.

Deriving a key from subscription data and `MK`

The main advantage of deriving a key from non-secret (subscription) data under a master key is that such derivable keys do not need to be stored and that the back-up of the subscriber keys is reduced to the back-up of the master key. No database containing secret information is thus required in the AC. When an authentication request comes from the VLR, the AC would just load the relevant data, say the IMSI, into the algorithm and derive the individual subscriber authentication key K_i from this data using the top secret master key `MK`. This method has a few undesirable effects if it is not managed with extreme care from both an administrative as well as a security point of view. The main problem is of course to keep the very secret key `MK` secret. Anyone coming into possession of this key could (if he knows the method of deriving K_i and the algorithms `A3` and `A8`) compromise every SIM card issued under `MK`. The method can also lead to the production of ”identical” SIMs. If the same IMSI has been used by mistake to generate the keys for two SIMs, these SIMs will be identical from a security point of view, i.e. they contain the same IMSI and K_i. One can avoid the risk of producing identical SIMs by combining subscription data with random data.

Getting authentication keys from random generator

Using a random number generator to produce the subscriber authentication keys ensures that all strings consisting of 128 bits are equally likely. This advantage cannot be achieved by an algorithm using IMSIs as an input. As there is no ”link” between the subscription and the authentication key, all keys have to be stored in a database of the AC and have to be backed-up at a physically different location. To protect the keys against unauthorized reading in the AC they have to be stored in an encrypted form. The key (or keys) used for decrypting the subscriber authentication keys is clearly very sensitive.

Example for Signalling in GSM

We will now exemplarily discuss the protocol for the allocation of a new Temporary Mobile Subscriber Identity (TMSI). It presents a typical signalling sequence over the radio interface. All transmitted messages and commands correspond to those of the SS7 (Signalling System Nr.7), thus to those in the fixed network. If necessery, they are adequately modified in the network and passed to the relevant interfaces.

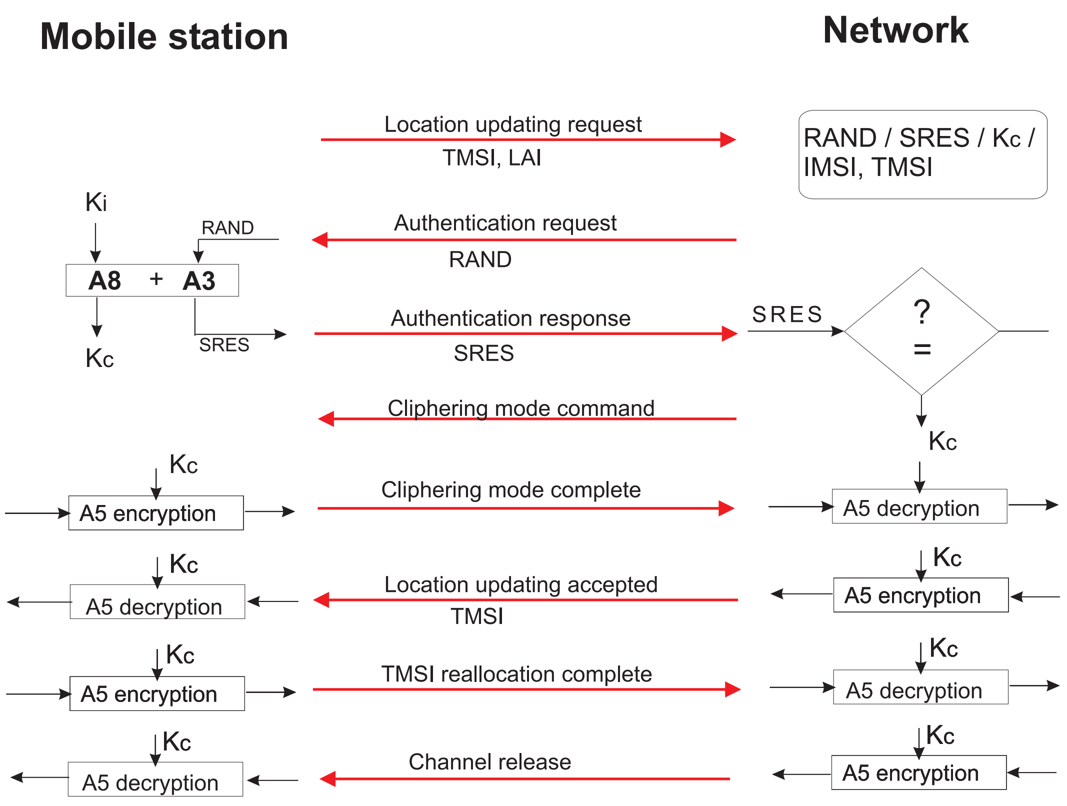


Fig. 8.2-10: Example for signalling in GSM (protocol for allocation of new TMSI)

Protocol for allocation of new TMSI

When a subscriber (user) wants to change his temporary pseudonym TMSI, he announces his wish to the network, specifying thereby its current TMSI and LAI (see Fig. 8.2-10). The network requests his authentication and sends him the challenge RAND. If the user is legitimate, he can sign the challenge with the authentication key K_i and then send the response SRES with the ”authentication response message” to the network. He can also generate his session key K_c. The network verifies the signed response SRES and if this was successful, it asks the subscriber with the command ”Ciphering mode” to change into the mode, in which all messages and commands are encrypted. The subscriber changes the mode and acknowledges the change with the ”Ciphering Mode Complete” message. The network can decrypt this message, and allocate a new TMSI, which it sends together with the ”Location updating accepted” message. The subscriber acknowledges the reallocation of the TMSI. After that, the network releases the channel. Herewith the procedure of TMSI reallocation is completed. Note that all messages after the ”ciphering mode command message” are sent encrypted.

Distributed Databases

In the typical signalling protocol between a user and the network presented above, we assume that all information required for the authentication and encryption processes resided in one place in the network. But in reality, the data is distributed physically over several databases. Messages, information and parameters between the distributed databases are transfered with strictly defined protocol sequences.

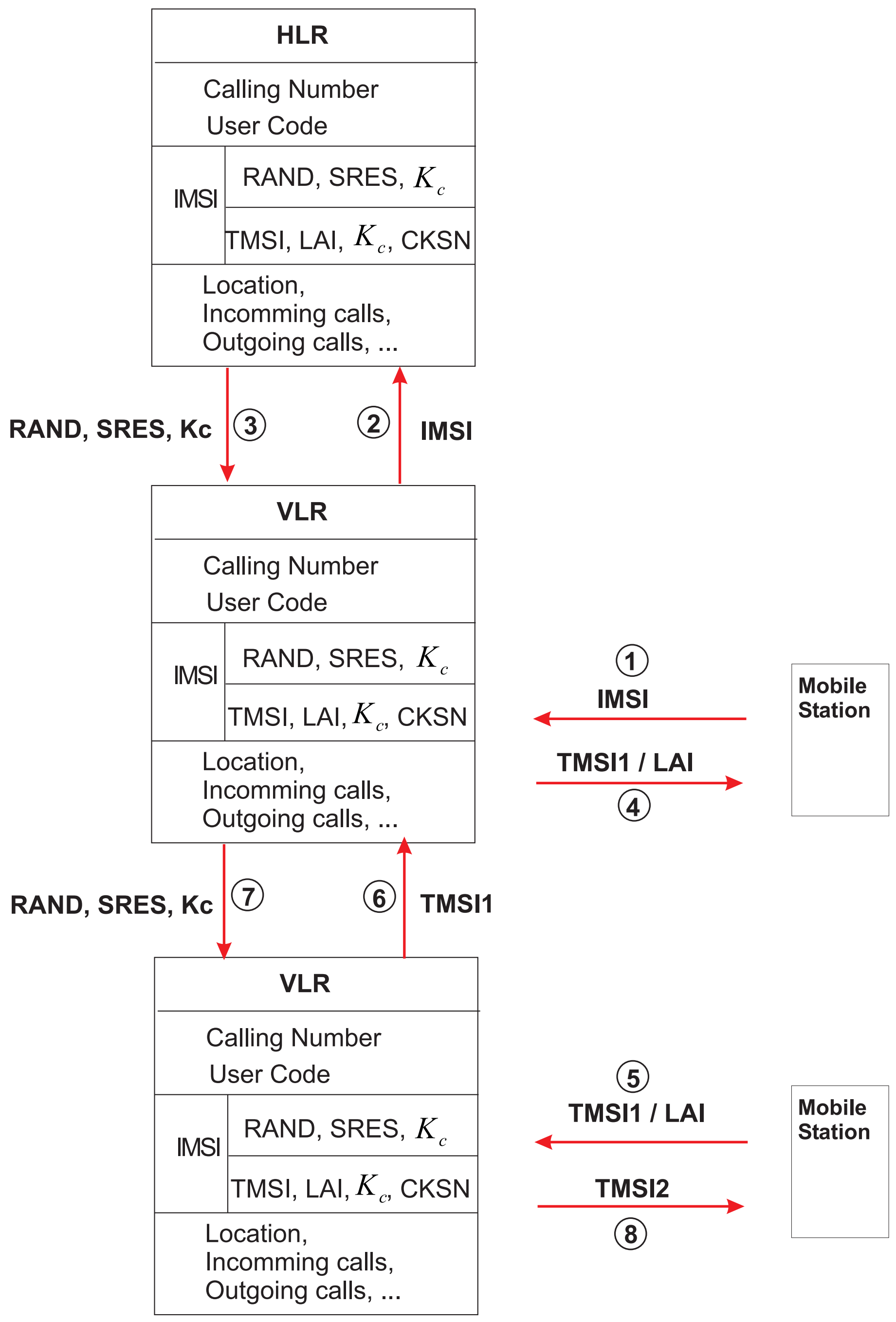


Fig. 8.2-11: Distributed databases in GSM

Accessing a "foreign" network

To illustrate the principle of distributed databases, we now describe the process when a user wants to access a network other than his home network (see Fig. 8.2-11). A user can access the network from anywhere with his IMSI. The VLR of the visited radio network part requests the necessary data from the Home Location Register (HLR) and receives sets of random bit streams RAND, associated responses SRES and session keys K_c, which it stores for the time of the visit. The VLR assignes a temporary identity TMSI1 to the user, which is transmitted to him encrypted. In Fig. 8.2-11 the message sequence is displayed in a very simplified manner. This also applies for the message sequences between the databases (HLR-VLR and VLR-VLR), which belong to the SS7 protocol.

When the user moves to another network location area, also belonging to the ”foreign” network, he can authenticate himself with his pseudonym TMSI1 and the LAI of the last network location area. The responsible VLR of the new network location area requests the required data ($RAND$, $SRES$, K_c) from the issuer of the pseudonym TMSI1, the VLR1. The VLR2 generates a new pseudonym TMSI2 for the user, and sends it encrypted together with K_c to him.

Note, if the user is outside of his home network, the visited network requests a set of (RAND, SRES, K_c) from the home network, and thus allows the visited network to communicate with the mobile station without gaining access to the algorithms A3 and A8.

As the SIMs and their contents (including A3 and A8) are controlled by the respective network operators, this structure leaves room for national and business policy enforcement. The algorithm A5, on the other hand, has to be supported by all networks and end devices in order to interoperate properly.

Trust relations

GSM offers confidentiality, subscriber authentication, and subscriber identity confidentiality. The security mechanisms are applied to all traffic, including short messages and control information. Note, that the security mechanisms are only defined for the air interface (mobile device to base station), i.e., **security of transport through fixed networks behind the base stations is left to the network providers**.

Fixed part of the GSM behind the BSs

Although the algorithms are not published officially, one widely employed implementation of A3/A8 called COMP128 and an algorithm compatible to A5 have been published. Attacks to the algorithm have also been published. This has lead to some uncertainty with regard to the actual strength of encryption and authentication of productive GSM networks.

Trust relations

Users and service providers relying on the GSM security have to trust the network providers regarding following aspects:

- Privacy of information and keying material, and
- Proper choice of the algorithms A3 and A8.

Given the security infrastructure described above, ”pure” GSM should only be used for telephony services and application with low sensitivity, like public information services. Examples for such services include general information (e.g. weather forecasts, sport results, nearest restaurant) and non-sensitive personalized financial information (e.g. stock information). Sensitive applications like account statements or financial transactions should employ additional security mechanisms as described in the following section.

8.2.6 SIM Application Toolkit

Purpose of SIM Application Toolkit

A **SIM application toolkit** is a GSM specification which defines an interface between GSM handsets and Subscriber Identity Modules (SIMs). The SIM Application Toolkit allows applications stored on the SIM to communicate through the handset with the user and the network. In other words, applications on the SIM can use the handset as an I/O device with the help of the SIM Application Toolkit, which, among others, includes the following functionalities: display of text and menus, receiving input from the keypad, sending and receiving short messages, set up of calls, and communication with a secondary smart card (for dual-slot handsets). Although a SIM Toolkit application can be any kind of application running on a given SIM, there are **standardisation efforts** underway within ETSI to define an applicaton programming interface for higher-level languages (SIM API).

Standardisation efforts for SIM APIs

End-to-end security

In contrast to basic GSM security, SIM Application Toolkit allows a **secure end-to-end connection** between the subscriber and a service or content provider, such that messages can be encrypted, e.g., between a SIM and a banking server. This makes security independent of limitations of the GSM algorithms.

If a dual-slot handset is used, the SIM Application Toolkit can make use of a secondary cryptographic smart card. This is a useful feature if, for example, users already have a standardised signature card. The SIM application can then coordinate the display of data to be signed, the PIN generation and the actual signature generation on the secondary card.

Trust relations

Although SIM Application Toolkit allows an end-to-end security, the GSM provider is still involved in the **trust relations** because he usually owns the SIM. All data put on the SIM, including application and keys, is actually under the control of the network provider. The communication parties have to trust the network provider and the card manufacturer to the effect that they do not misuse or leak the (potential) knowledge of information stored on the SIM. There is no trust necessary concerning intermediate networks, as in the case of ”pure” GSM security.

The SIM Application Toolkit is most suitable for sensitive, personalized services, such as banking and brockerage. Security mechanisms can be agreed upon individually per application. One basic application which is useful as building block for many solutions is a signature application. In such an application, the SIM contains the private signing key and some application logic which controls the signature process. It receives short messages which contain the document to be signed, displays the content to the user, and generates the signature on user demand. The signatures can then be sent back to the source of the document, or to a different recipient. In this way the handset becomes a general-purpose and higly secure signature device. The signature process can be triggered by any kind of external application (e.g. web application).

8.3 Security in UMTS

The abbreviation UMTS stands for Universal Mobile Telecommunications System. It is a third generation mobile radio system developed within the EU programmes RACE[67] (1989-1994) and ACTS[68] (1995-1998), in cooperation with ETSI[69]. In the meantime, the standardization process for UMTS has moved from ETSI to **3GPP** (*Third Generation Partnership Project*). 3GPP is a standardization organization formed by various regional organizations and other related bodies from different countries[70].

Developement of UMTS

The UMTS technology integrates the different mobile communications systems available worldwide today, such as mobile phones and satellite radio systems, into one system. It offers a common air interface covering all fields of applications. Therefore, users can transmit voice, text, data and images over one UMTS connection with a relatively high transmission capacity of up to 2 Mbit/s.

8.3.1 Architecture of UMTS

Domains in UMTS

From the physical point of view, UMTS is divided into different domains. Fig. 8.3-1 shows the architecture of UMTS. A basic architectural split is between the user equipment domain and the infrastructure domain, which is further divided into the access network domain and core network domain.

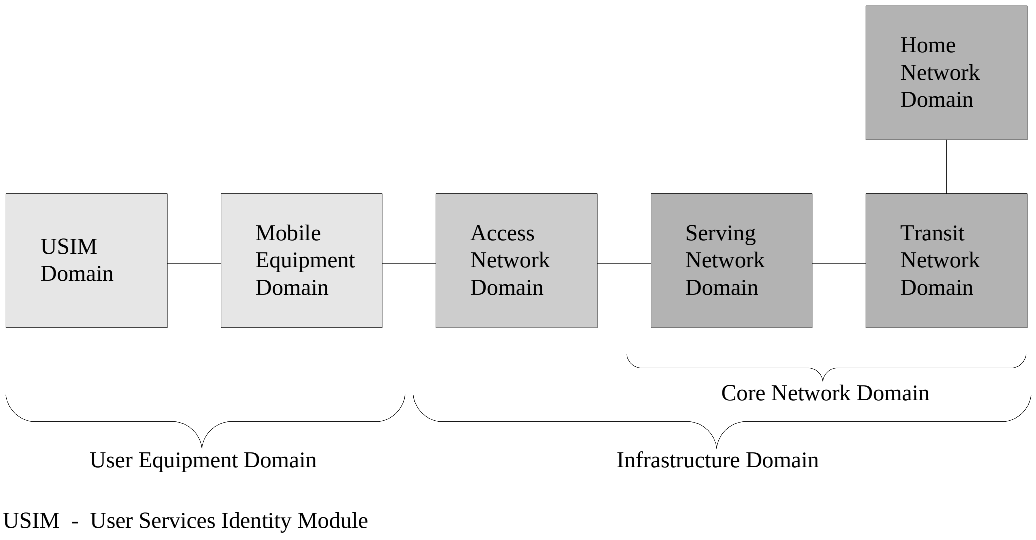


Fig. 8.3-1: UMTS domains and interfaces.

67 Research and Development in Advanced Communications Technologies in Europe.

68 Advanced Communications Technologies and Services.

69 European Telecommunications Standards Institute.

70 ETSI from Europe, ARIB (Association of Industries and Businesses) and TIA (Telecommunication Industry Association) from Japan, TTA (Telecommunication Technology Association) from Korea, CWTS (China Wireless Telecommunication Standard) from China, and T1 from North America.

User Equipment Domain

User equipment domain

This domain is split into the *User Services Identity Module* (**USIM**) domain and *Mobile Equipment Domain*. The Mobile Equipment Domain performs radio transmission and contains applications. The USIM domain contains all information and procedures used for encryption of the communication and for authentication of a terminal to the network. The USIM is an application implemented in the *UMTS IC card* (**UICC**). As in the GSM network, the user identifies himself to the UICC (for using the USIM application) by providing his *Personal Identification Number* (**PIN**)

Access Network Domain

Access network domain

This domain provides the user with a mechanism to access the UMTS core network. The access to the core network can be realized by using the **UTRAN** (*UMTS Terrestrial Radio Access Network*) technique, the GSM-BSS technique as used in the GSM network, or satellite communications.

Core Network Domain

Core network domain

This domain is an integral platform that can consist of different transport networks such as Internet, GSM networks, etc. which are linked together via network gateways. The core network domain is further divided into three domains:

Serving network domain

- **Serving network domain**: This domain is the part of the core network domain to which the access network domain is connected. It is responsible for routing calls and transport of user data from source to destination, according to the present user access point.

Home network domain

- **Home network domain**: It works like the HLR[71] and provides core network functions that relate to a fixed location (the location where the user is registered). The USIM is related by subscription to the home network domain. The latter is, therefore, responsible for management of subscription information and for handling home specific services, potentially not offered by the serving network.

Transit network domain

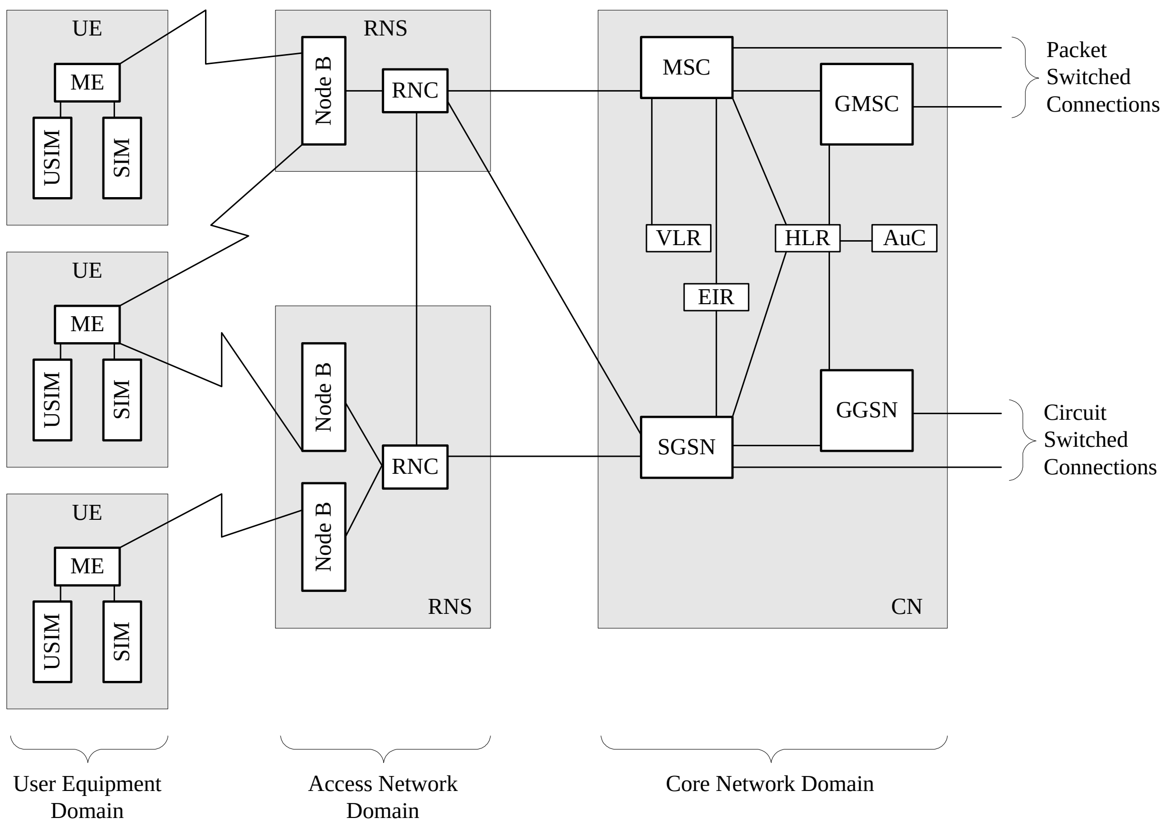
- **Transit network domain**: This domain is the core network part located on the communication path between the serving network domain and the remote party. It intervenes only if the remote (called) party is not in the same network as the calling party.

UTRAN as access network

Fig. 8.3-2 shows the UMTS architecture when UTRAN is used as access network. The UTRAN consists of *Radio Network Subsystems* (**RNS**), each of which is connected to the *core network* (**CN**). An RNS consists of an *Radio Network Controller* (**RNC**) and one or more **Node B**. In a UMTS environment, the base stations are called Node B. A Node B is a logical unit responsible for the radio transmission in one or more cells. It communicates with the RNC using the Iub interface. The task of the RNC is to controll the use and the integrity of the radio resources (e.g.

71 See Section 8.2 "Security Mechanisms in GSM Networks".

frequencies). The RNC is also responsible for the handover decisions that require signalling to the user equipment.



AuC	- Authentication Center	Node B	- In UMTS the base stations are called Node B
CN	- Core Network	RNC	- Radio Network Controller
EIR	- Equipment Identity Register	RNS	- Radio Network Subsystem
GGSN	- Gateway GPRS Support Node	SGSN	- Serving GPRS Support Node
GMSC	- Gateway Mobile-Services Switching Center	SIM	- Subscriber Identity Module
GPRS	- General Packet Radio Service	UE	- User Equipment
HLR	- Home Location Register	USIM	- User Services Identity Module
ME	- Mobile Equipment	VLR	- Visitor Location Register
MSC	- Mobile Switching Center		

Fig. 8.3-2: UMTS architecture with UTRAN (UMTS Terrestrical Radio Access Network) as access network.

The core network is logically divided into a *Circuit Switched Domain* (**CSD**) for channel-switched communication and a *Packet Switched Domain* (**PSD**) for packet switched communication (Fig. 8.3-2)[72]. In the CSD, resources are reserved during connection setup and not released until termination, whereas in the PSD packets can be sent independently of one another without the need to reserve resources in the network.

72 For more information see [Kaderali00b].

The MSC, the **GMSC** (*Gateway Mobile-services Switching Centre*) and the VLR are part of the CSD. The EIR and the HLR (resp. AuC) are used jointly from both domains. Note that the functionalities of the entities MSC, VLR, EIR, HLR and Auc have already been discussed in Section 8.2. The *Serving GPRS Support Node* (**SGSN**) represents the *General Packet Radio Service* (**GPRS**) switching centre in analogy to the MSC. GPRS is a technology developed for the evolution of the GSM system towards third generation mobile communication systems. This technology represents the first implementation of packet switching within GSM, which is essentially a circuit-switched technology. The *Gateway GPRS Support Node* (**GGSN**) serves as the interface towards external public data networks or other public land mobile networks.

8.3.2 Security Principles of UMTS

The security principles of UMTS are based on three aspects [DU99]:

1. Security elements within GSM and other second generation systems that have proved to be useful and robust, shall be adopted for UMTS. Thus, the following security elements of second generation have to be – as a matter of principle – retained for UMTS:
   - Subscriber to network authentication.
   - Use of a SIM card as a terminal independent security module, consisting of removable hardware.
   - Authentication of subscriber to the SIM.
   - Subscriber identity confidentiality.
   - Radio interface encryption.
   - The operation of security features (for example, encryption of communication transferred over the air interface) is independent of user assistance, that means the user does not need to do anything for the security features to be in operation.
   - SIM application toolkit security features provide a secure application layer (end-to-end security) between the SIM and the network.
   - The authentication procedure is performed by the serving network. Nevertheless, the Home Environment (HLR in which the subscriber is registered) has minimal trust in the serving network, hence no permanent user-individual keys are transferred from the user's Home Environment to the serving network.
2. The UMTS security will improve the security of the second generation systems. That is, the UMTS security will address and correct the security weaknesses in second generation systems. Some addressed weaknesses are:

- Cipher keys and authentication data (`RAND`, `SRAND`, K_c) are transmitted in clear mode (e.g. unencrypted) between and inside networks, e.g. between HLR and VLRs.
- False BTS attack: This attack can be performed by an attacker who is in possession of an ”IMSI catcher” ([Stepping02], [Fox02a]). So, he can play the role of a (falsified) base station and prevent the user from establishing a connection to the appropriate (true) base station. This happens because the user of the MS receives the highest reception level from the IMSI catcher when he attempts to connect to the network. Fig. 8.3-3 shows the false BTS attack, where the false base station is an IMSI catcher. On the other hand, the IMSI catcher acts as a usual mobile user to the network and forwards the outgoing calls generated by the victim.

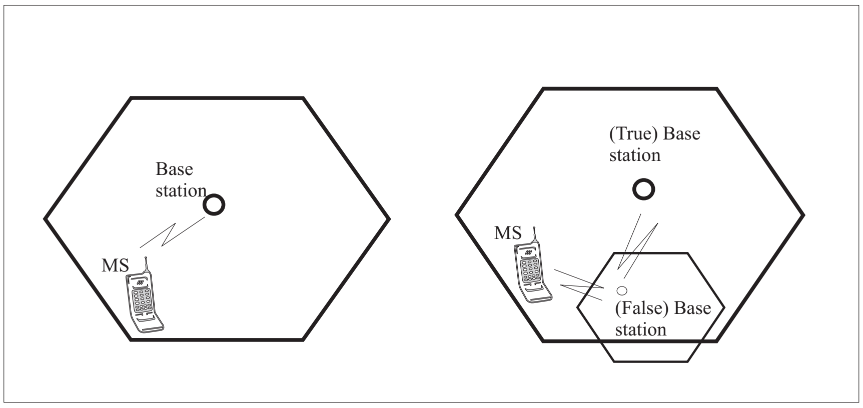


Fig. 8.3-3: False base station attack (right) and usual communication (left).

- In GSM, user and signalling data are not encrypted between the BTS and the BSC. The encryption over the air is terminated – earlier as necessary – at the BTS.
- In some cases, where user authentication using `RAND`, `SRES`, and `A3/A8` cannot be provided, user authentication can be done with the use of a previously generated cipher key. That is, the encryption leads to an implicit user authentication. Some networks, however, do not support encryption. Subsequently, these networks cannot provide implicit user authentication, too. This makes some attacks, such as channel hijacking, possible.
- Data integrity is not provided. Data integrity is one means to prevent false base station attacks if encryption is not supported by the network.
- The home environment has no knowledge or control of how a serving network uses authentication parameters (`RAND`, `SRES`, ...) for home environment subscribers roaming within that serving network.

3. New services offered by the UMTS technology will be protected by using new security features. For example, the UMTS terminal can be used as a platform for e-commerce and other applications. So, the terminal may support personal authentication using biometric methods.

Regarding the security weaknesses in GSM listed above, we now introduce the new security features added to the UMTS system.

8.3.3 Authentication and Key Agreement (AKA)

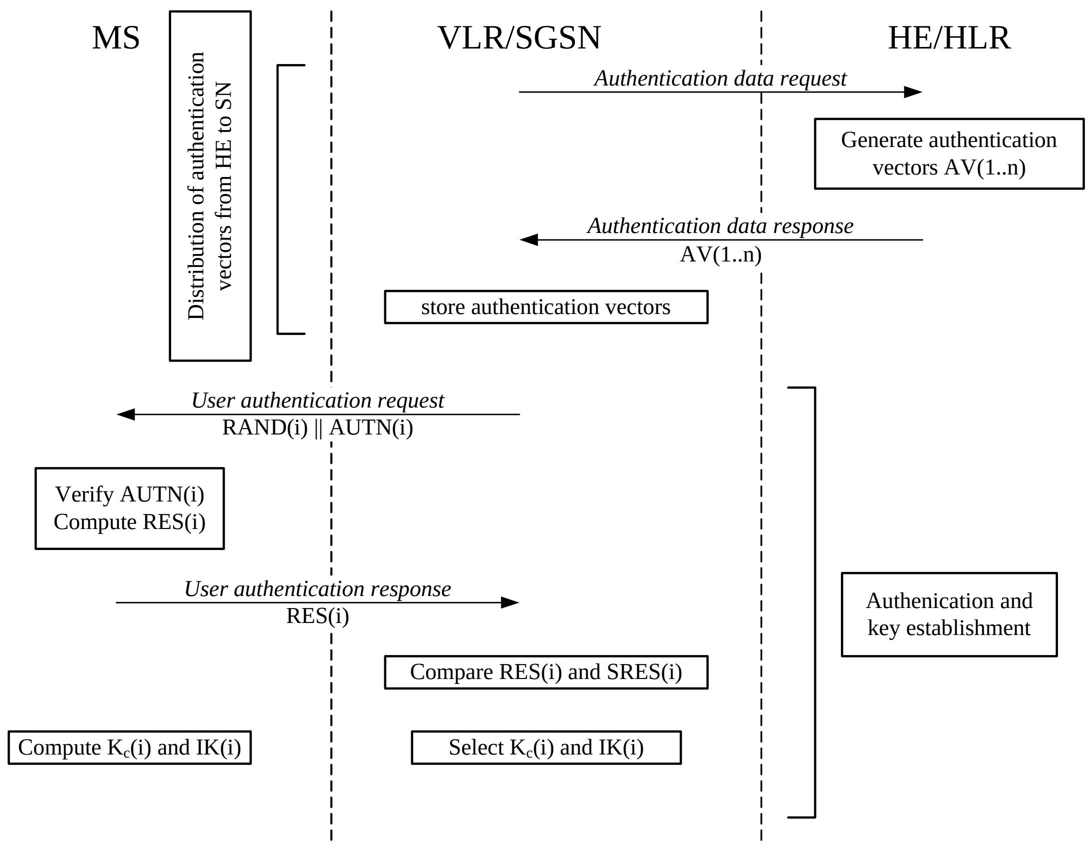
The UMTS authentication and key agreement protocol has been designed in such a manner that the compatibility with GSM is maximized and a migration from GSM to UMTS is easily possible. Analog to GSM, UMTS uses a challenge-response protocol for authentication. However, there are significant enhancements to this protocol in order to achieve additional protocol goals [Puetz01a]:

- Authentication of the home environment to the user
- Agreement of an integrity key (`IK`) between the user and the serving network. That implies that the integrity of data and signalling information is supported.
- Mutual assurance of freshness of the agreed cipher key (K_c) and integrity key (`IK`) between the serving network and the user. Replay attacks are not possible.

An overview of the AKA protocol is shown in Fig. 8.3-4. On receipt of an authentication request from the VLR/SGSN, the HE/HLR (resp. AuC) sends an array of n authentication vectors[73] to the VLR/SGSN. Each authentication vector consists of a random number `RAND`, an expected response `SRES`, a cipher key K_C, an integrity key `IK`, and an authentication token `AUTN`. How these authentication vectors are generated, will be explained later.

In an authentication exchange, the VLR/SGSN selects the next (the i-th, $1 \leq i \leq n$) authentication vector from the ordered array and sends the parameters `RAND(i)` and `AUTN(i)` to the user. The USIM (i.e. the user) checks whether `AUTN(i)` can be accepted, and if so, produces a response `RES(i)` which is sent back to the VLR/SGSN. The latter compares it with its `SRES(i)` value. The communication is set forth only if both are equal. The USIM now computes the keys K_c and `IK` which are used for encryption and integrity of data transferred over the air interface.

73 The equivalent of the GSM triplet `RAND`, **SRES**, and K_C.


MS
VLR/SGSN
HE/HLR
Distribution of authentication vectors from HE to SN
Authentication data request
Generate authentication vectors AV(1..n)
Authentication data response
AV(1..n)
store authentication vectors
User authentication request
RAND(i) || AUTN(i)
Verify AUTN(i)
Compute RES(i)
User authentication response
RES(i)
Authenication and key establishment
Compare RES(i) and SRES(i)
Compute K_c(i) and IK(i)
Select K_c(i) and IK(i)

AUTN	- Authentication Token	MS	- Mobile Station
AV	- Authentication Vector	RAND	- Random Number
GPRS	- General Packet Radio Service	RES	- Response
HE	- Home Equipment	SGSN	- Serving GPRS Support Node
HLR	- Home Location Register	SN	- Serving Network
IK	- Integrity Key	SRES	- Expected Response
K_c	- Cipher Key	VLR	- Visitor Location Register

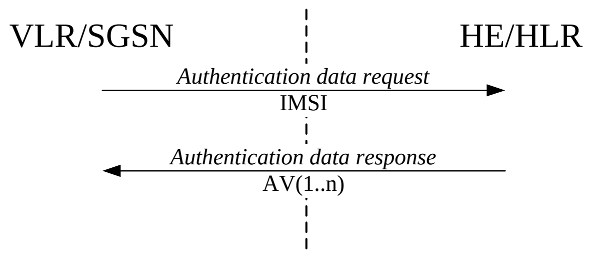
Fig. 8.3-4: Authentication and key agreement protocol

Generation of Authentication Vectors

It is assumed that the authentication centre AuC of the user's home environment and the user's USIM share a user specific secret key K_i with 128 bit length, certain message authentication functions f_1, f_2, and certain key generating functions f_3, f_4, f_5.

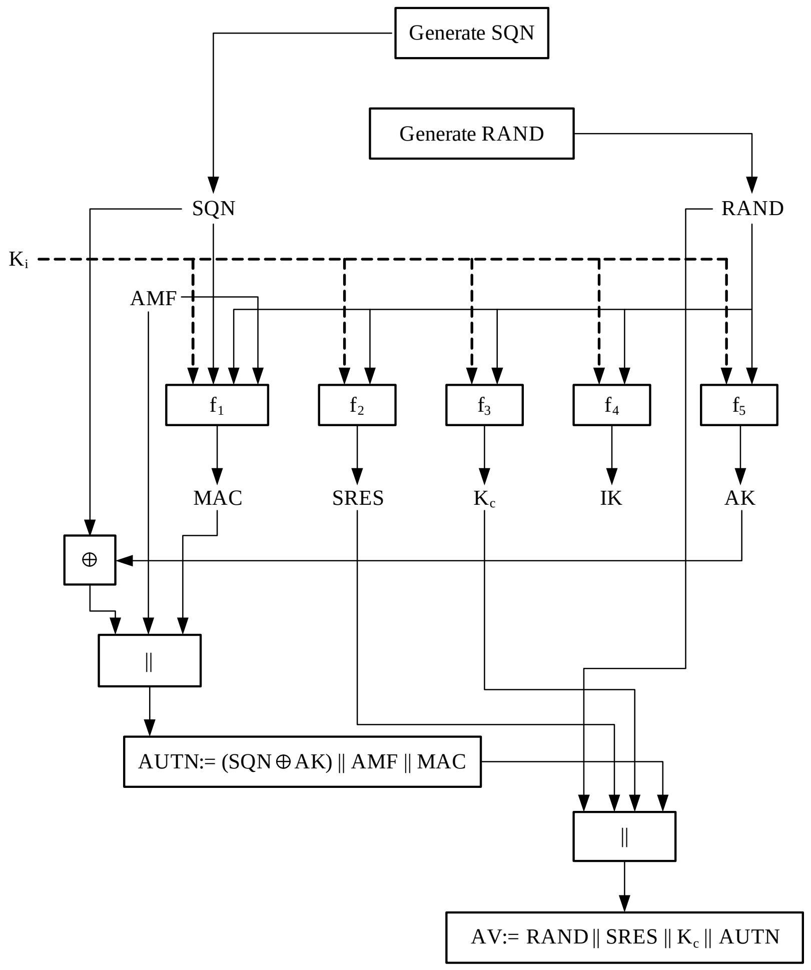
Upon receipt of an authentication data request, the HE/HLR (resp. AuC) starts with generating a fresh sequence number `SQN` and an unpredictable random number `RAND`. For each user the HE/HLR keeps track of a counter SQN_{HE}. Thereafter, the HE/HLR computes the following values by using the secret key K_i and an operator specific *Authentication Management Field* (**AMF**[74]):

74 `AMF` defines the operator-specific options used in the authentication processes, e.g the lifetime restriction of cipher and integrity keys.



AV	- Authentication Vector	IMSI	- International Mobile Subscriber Number
GPRS	- General Packet Radio Service	SGSN	- Serving GPRS Support Node
HE	- Home Equipment	VLR	- Visitor Location Register
HLR	- Home Location Register		

Fig. 8.3-5: Distribution of authentication Data from HE/HLR to VLR/SGSN



⊕	- Bitwise exclusive-OR operation	K_c	- Cipher Key
||	- Concatenation of two ore more operands	K_i	- User Specific Key
AK	- Anonymity Key	MAC	- Message Authenication Code
AM	- Authentication Management Field	RAND	- Random Number
AUTN	- Authentication Token	SQN	- Sequence Number
AV	- Authentication Vector	SRES	- Expected Response
IK	- Integrity Key		

Fig. 8.3-6: Generation of authentication vectors

- A message authentication code `MAC =` $f_{1_{K_i}}$ $(SQN||RAND||AMF)$[75].
- An expected response $SRES = f_{2_{K_i}}(RAND)$.
- A cipher key $K_c = f_{3_{K_i}}(RAND)$.
- An integrity key $IK = f_{4_{K_i}}(RAND)$.
- An anonymity key $AK = f_{5_{K_i}}(RAND)$.

The authentication token can be constructed as:

$$AUTN = (SQN \oplus AK)||AMF||MAC.$$

The *Anonymity Key* (**AK**) is used to hide the sequence number which may expose the identity and location of the user.

As described in the AKA protocol, the VLR/SGSN sends the values `AUTN(i)` and `RAND(i)` in the user authentication request (see Fig. 8.3-4). Upon receipt of `AUTN(i)` and `RAND(i)`, the USIM proceed as shown in Fig. 8.3-7. Note that in the figure the index i is ignored.

The USIM computes the anonymity key $AK = f_{5_{K_i}}(RAND)$ and retrieves the sequence number

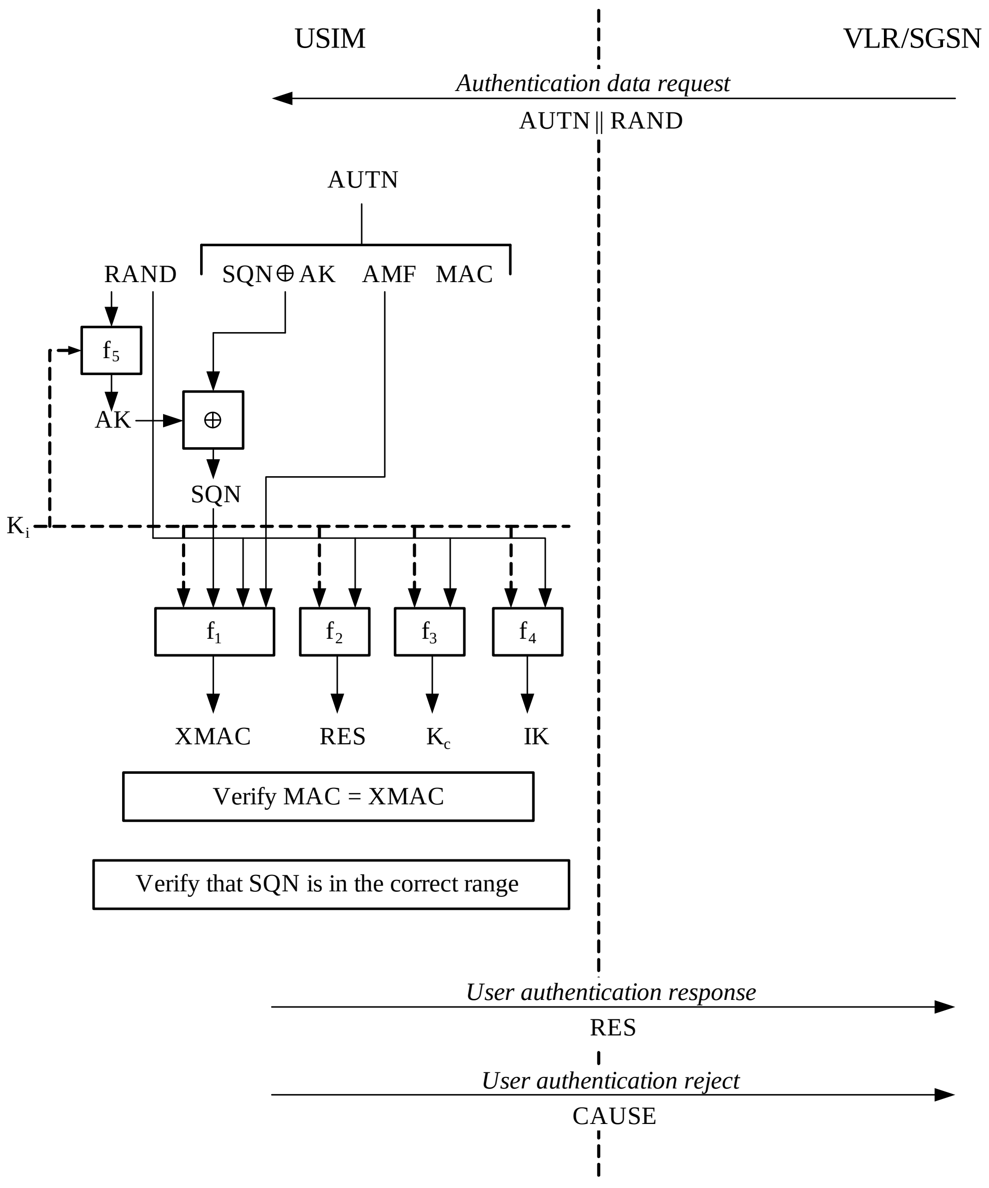
$$SQN = (SQN \oplus AK) \oplus AK.$$

Next, the USIM computes

$$XMAC = f_{1_{K_i}}(SQN||RAND||AMF)$$

and compares it with the MAC included in the token `AUTN`. If they are different, the user sends a *user authentication reject* message back to the VLR/SGSN with an indication of the cause and then abandons the procedure. Otherwise, the USIM verifies that the received sequence number SQN is in the correct range. If the USIM considers that the sequence number is not in the correct range. It sends a *synchronization failure* message back to the VLR/SGSN, an integrity-protected information about an acceptable sequence number, and abandons the procedure. The VLR/SGSN requests fresh authentication vectors by forwarding the *synchronization failure* message to the HE.

75 || indicates that the next term is appended (concatenated) to the one before. Thus $SQN||RAND$ means that `RAND` is appended to `SQN`


USIM
VLR/SGSN
Authentication data request
AUTN || RAND
AUTN
RAND
SQN⊕AK
AMF
MAC
f5
AK
⊕
SQN
Ki
f1
f2
f3
f4
XMAC
RES
Kc
IK
Verify MAC = XMAC
Verify that SQN is in the correct range
User authentication response
RES
User authentication reject
CAUSE

⊕	- Bitwise exclusive -OR operation	RAND	- Random Number
AK	- Anonymity Key	RES	- Response
AMF	- Authentication Management Field	SGSN	- Serving GPRS Support Node
AUTN	- Authentication Token	SQN	- Sequence Number
IK	- Integrity Key	USIM	- User Services Identity Module
K_c	- Cipher Key	VLR	- Visitor Location Register
K_i	- User Specific Key	XMAC	- Expected MAC
MAC	- Message Authenication Code		

Fig. 8.3-7: User authentification in the USIM

In the case where the USIM verifies that the sequence number lies in the correct range, it computes $RES = f_{2_{K_i}}(RAND)$ and includes this parameter in a user authentication response back to the VLR/SGSN (see Fig. 8.3-4). Finally, the USIM computes the cipher key $K_c = f_{3_{K_i}}(RAND)$ and the integrity key $IK = f_{4_{K_i}}(RAND)$. The fact that `MAC = XMAC` ensures that the random challenge `RAND` sent by VLR/SGSN is really generated by the user's HE, and that the actual VLR/SGSN is trusted by the HE to deal correctly with authentication vectors. Thus, the authentication of the network is reached. Furthermore, checking that `SQN` lies in the correct range implies the freshness of the keys K_c and `IK`, because

SQN and RAND (from which the keys K_c and IK are generated) are jointly integrity protected by the MAC. In other words, any attempt by the VLR/SGSN to reuse a quintet more than once will be rejected by the MS.

Once the VLR/SGSN receives the value of RES, it compares it with the expected value SRES. If they are equivalent, the user is authenticated by the network.

If the AKA protocol is carried out properly, the VLR/SGSN is in possession of a cipher key K_c and an integrity key IK. These keys are then transferred to the RNC when needed. Note that there are special cipher/integrity keys for circuit-switched domains (CSD), and other cipher/integrity keys for packet-switched domains (PSD). For simplicity, we use the same notation for the cipher key K_c and the integrity key IK, independently from the used domain. Now, user data and some signalling information can be confidentiality protected with the use of the standard algorithm f_8.

8.3.4 Encryption of the Communication

The f_8 algorithm is a stream cipher used to encrypt the plaintext by applying a keystream to it. Therefor a bit per bit binary addition of the plaintext and the keystream is used (see Fig. 8.3-8). The decryption of the plaintext is obtained by an XOR addition of the ciphertext and the same keystream. The encryption occurs between the *Radio Network Center* (RNC) and the *User Equipment* (UE)[76]. Recall that in GSM, the encryption takes place only between the mobile unit and the BTS (and not BSC).

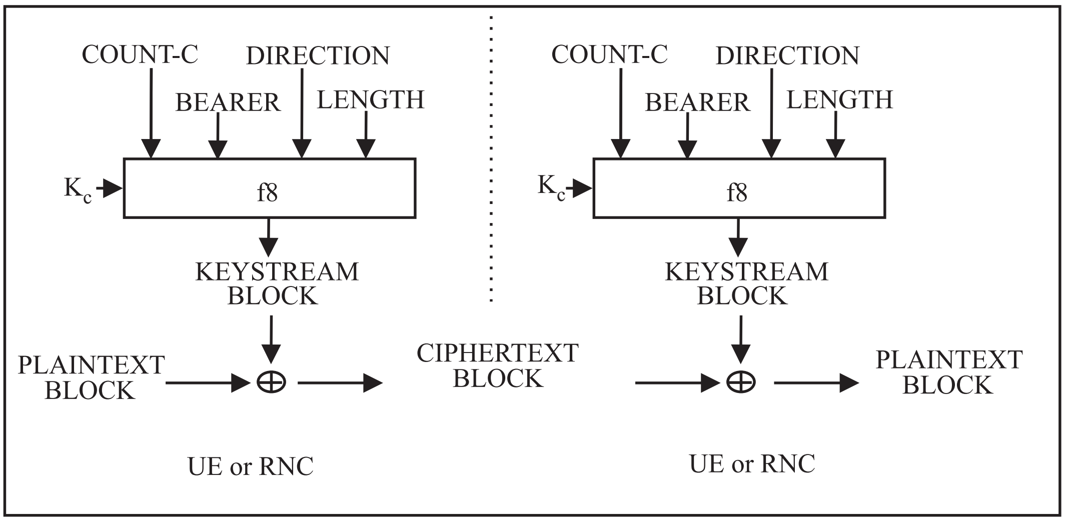


Fig. 8.3-8: Encryption/Decryption of the communication

The input parameters to the f_8 algorithm are:

76 More precisely, between RNC and the mobile equipment ME which is a part of the UE.

- The cipher key K_c (128 bit),
- `COUNT-C` (32 bits), which represents the hyper-frame number as a time-variant parameter [Stepping02],
- `BEARER` (5 bits), which identifies the physical channel (”radio bearer”) used to transmit the data,
- `DIRECTION` (1 bit), which takes the value 0 for messages sent from UE to RNC and 1 for messages sent from RNC to UE. It is given as input to f_8 in order to avoid that the keystreams for the uplink and downlink would use the same input parameter values.
- `LENGTH` (16 bits), which determines the length of the required keystream block, and

The f_8 algorithm is based on the KASUMI cipher [3GP02c]. KASUMI is a block cipher that produces a 64-bit output from a 64-bit input under the control of a 128-bit key K_i [3GP02d]. f_8 is a stream cipher that encrypts/decrypts blocks of data between 1 and 20000 bits in length. It uses the block cipher KASUMI in OFB mode (see Fig. 8.3-9).

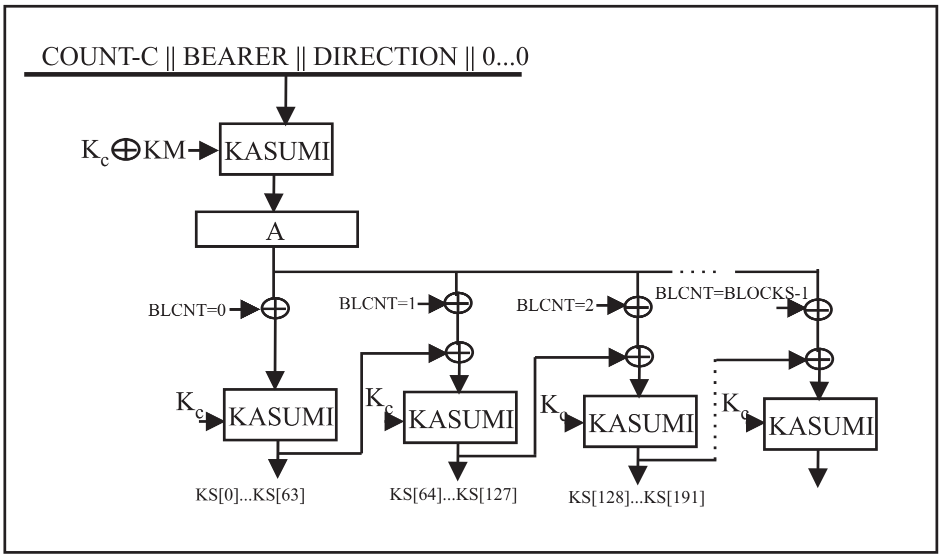


Fig. 8.3-9: KASUMI cipher used in OFB mode.

Before generating the keystream bits, the f_8 algorithm must be initialized as follows (see Fig. 8.3-9):

- The 64-bit register `A` is set to $COUNT||BEARER||DIRECTION||0\dots0$, with the right most 26 bits set to 0,
- `BLKCNT` is set to zero. Note that `BLKCNT` is specified as a 64-bit counter,
- The 128-bit key modifier `KM` is set to `55555555555555555555555555555555`,

– KSB_0 is set to zero. Note that KSB_i is the i^{th} block of keystream produced by the keystream generator, where each block of keystream comprises 64 bits,

– One operation of KASUMI is then applied to the register A, using a modified version of the confidentiality key: $A = KASUMI[A]_{K_c \oplus KM}$.

The f_8 algorithm is now initialized and ready to generate the keystream bits. The plaintext/ciphertext to be encrypted/decrypted consists of `LENGTH` bits ($1 \leq LENGTH \leq 20000$). Between 0 and 63, the least significant bits are then discarded from the last block of the keystream depending on the total number of bits required by `LENGTH`.

So let `BLOCKS` be equal to ($LENGTH/64$) rounded up to the nearest integer. For instance, if $LENGTH = 128$ then $BLOCKS = 2$, if $LENGTH = 129$ then $BLOCKS = 3$. To generate each keystream block `KSB`, the following operation is performed:

$$KSB_n = KASUMI[A \oplus BLKCNT \oplus KSB_{n-1}]_{K_c},$$

for each integer n with $1 \leq n \leq BLOCKS$, and $BLKCNT = n - 1$.

8.3.5 Integrity Protection

To protect the integrity of sensitive signalling information which is sent between the MS and the network, an algorithm called f_9 is used. Like thef_8 algorithm, f_9 is applied between the *Mobile Equipment* (ME) and the *Radio Network Center* (RNC). Fig. 8.3-10 illustrates the use of the integrity algorithm f_9.

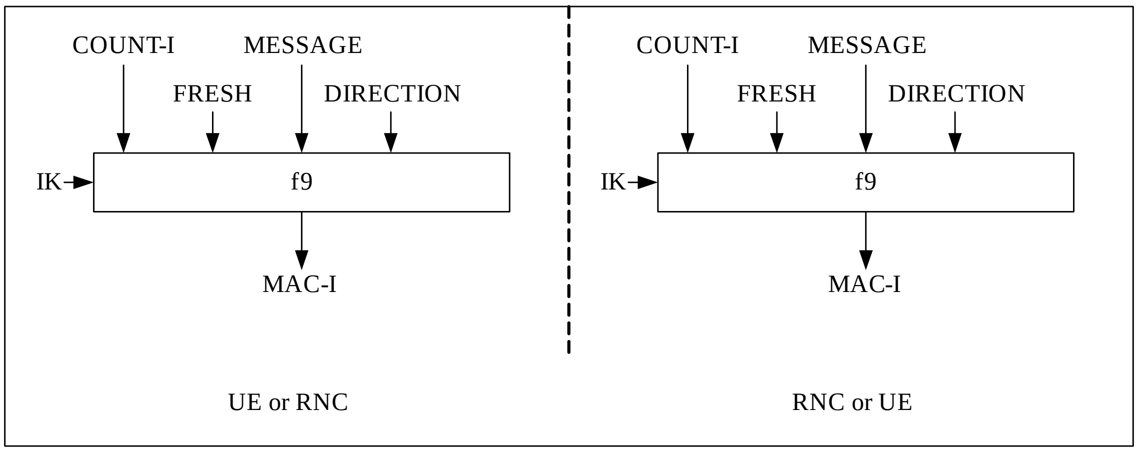


Fig. 8.3-10: Using the f_9 algorithm on a signalling message.

The input parameters to the f_9 algorithm are:

– The integrity key `IK` (128 bit),

– `COUNT-I` (32 bits): A sequence number to add a temporal dependency [Stepping02],

– `FRESH` (32 bits), a random value generated by the network. There is one `FRESH` parameter value per user and this value is used to protect the network against replay of signalling messages by the user,

– `MESSAGE`: The message to be protected,

– `DIRECTION` (1 bit), defined like in the previous section.

All these input parameters are used by f_9 to compute a message authentication code `MAC-I`. The `MAC-I` is appended to the message before sending it over the radio access link. The receiver computes `X-MAC` on the message – in the same way as `MAC-I` is computed – and compares it with `MAC-I`. If they are the same, the message is considered to be integrity protected.

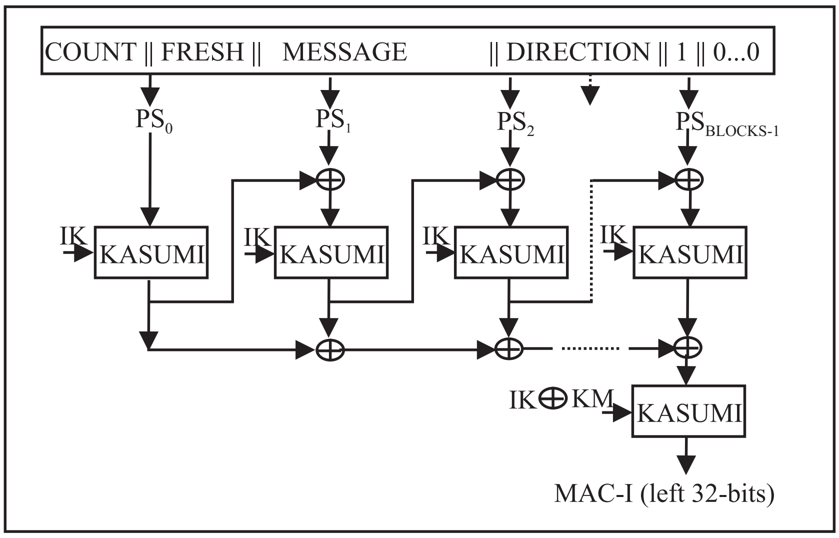


Fig. 8.3-11: Integrity algorithm f_9

The integrity algorithm f_9 is also based on the block cipher `KASUMI` which is used in the CBC mode to generate a 64-bit digest of the message input. Finally, the leftmost 32-bits of the digest are taken as the output value `MAC-I` (see Fig. 8.3-11). The input parameters `COUNT-I`, `FRESH`, `MESSAGE`, `DIRECTION`, a single bit 1 and between 0 and 63 '0' bits are concatenated so that the total length of the resulting string `PS` (*Padded String*) is an integral multiple of 64 bits. The Padded String `PS` is then split into 64-bit blocks PS_i, where $0 \leq i \leq BLOCKS - 1$ and $BLOCKS = (\text{bitlength of } PS)/64$. The PS_i are then processed according to Fig. 8.3-11 with `KM` set to AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA.

8.3.6 Securing the Core Network Domain

The core network domain of mobile radio systems is the part of the network which is independent of the radio interface technology. It is used for transporting user data as well as signalling commands [Puetz01a]. In 2G mobile systems like GSM the core network is not secured for two reasons:

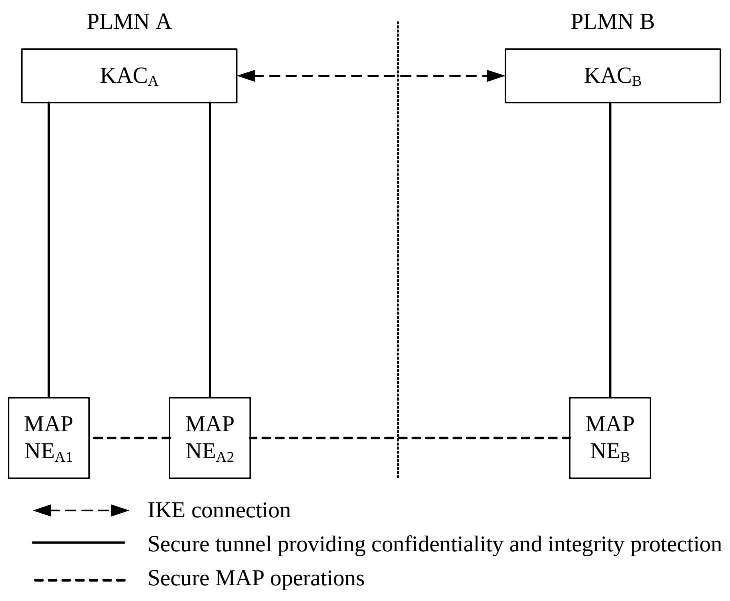
- The initial design goal of 2G mobile systems was to make them as secure as fixed network connections. This means that the security of 2G mobile systems depends only on the security provided at the radio interface.

- The core network is considered to be closed and hence not vulnerable to external attacks.

As mentioned in Course Unit 9, GSM uses the SS7 (Signaling System No. 7) protocol stack with *Mobile Application Part* (**MAP**) as an application on top for its signalling messages. According to the 2G security viewpoint, the security of SS7 network as transport system remains open to major compromises. Messages can be eavesdropped, altered, injected or deleted in an uncontrolled manner.

MAPsec

In 3G systems like UMTS, the security of the core network is taken more seriously into consideration. This means that security solutions must be found for both SS7 and IP based protocols. For this purposes, work has be done within the 3GPP in order to specify security mechanisms and procedures necessary to protect the MAP protocol. The complete set of enhancements and extensions to facilitate security protection for the MAP protocol is termed as **MAPsec** (see [3GP02a]). With networks using SS7-based transport, the MAP protocol is protected at the application layer. However, The MAP protocol can be protected at the network layer when the underlying transport network is IP-based. Fig. 8.3-12 shows the architecture used for MAPsec.



IKE	- Internet Key Exchange
KAC	- Key Administration Centers
MAP NE	- MAP Network Elements
PLMN	- Public Land Mobile Network

Fig. 8.3-12: The architecture used for MAPsec

In order to protect the communication over the core network, *Security Associations*[77] (**SA**) need to be established between the respective Mobile Application Part (MAP) network elements. Security Associations define, among other things, which keys and algorithms are to be useed to protect the MAP signalling. The necessary SAs for the use of MAPsec between networks are negotiated between the respective *Key Administration Centers* (**KACs**) of the networks. The KACs are newly defined network entities to support the MAPsec protocol. Each *Public Land Mobile Network* (**PLMN**) can have one or more KACs. The negotiated Security Associations hold PLMN-wide and is distributed to all network elements which implement MAP application layer security within the PLMN. These elements are denoted by MAP NE_X with respect to the PLMN X (see Fig. 8.3-12).

When a MAP Network Element (MAP NE) needs to establish a secure connection towards another MAP NE, it will request a MAPsec Security Association (MAPsec SA) from the Key Administration Centers (KAC) if it cannot find any appropriate MAPsec SA in its local *Security Association Database* (**SADB**). Note that before doing so, the MAP Network Element must consult its security policy database to check if MAP security mechanisms shall be applied towards the other MAP NE. The Key Administration Centers (KAC) then either provides an existing MAPsec Security Association or negotiate a new one, before returning it to the MAP NE. The negotiation of MAPsec Security Associations between the Key Administration Centers (KAC) is performed using the Internet Key Exchange (IKE) protocol (see Fig. 8.3-12) and with respect to the security policy defined in the KAC security policy database.

77 See chapter Internet Security Protocols in the course Network Security I.

A MAPsec Security Association (MAPsec SA) is valid for all MAP communications between the two PLMNs for which it was negotiated. That is, the same MAPsec SA shall be provided to all MAP Network Elements (MAP NE) in PLMN A for communication with MAP NEs in PLMN B.

The MAPsec protocol defines three different protection modes for securing the communication over the core network:

- Protection mode 0: No protection.
- Protection mode 1: Integrity and authenticity.
- Protection mode 2: Confidentiality, integrity and authenticity.

The secured MAP messages have the following structure:

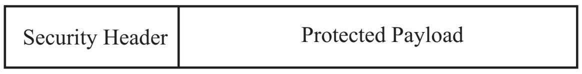


Fig. 8.3-13: Secured MAP message format.

In all three protection modes, the security header is transmitted in cleartext. For Protection Mode 0, the security header constitutes of the following data elements:

$$Security\,header = SPI||Original\,component\,Id.$$

For protection modes 1 and 2, the security header is a sequence of the following elements:

$$Security\,header = SPI||Original\,component\,Id||TVP||NE - Id||Prop.$$

The *Security Parameters Index* (**SPI**, 32 bit) identifies the used MAPsec SA. The *Original component Id* identifies the type of message to be transmitted. The *Time Variant Parameter* (**TVP**) is a 32 bit time-stamp used for replay protection of secured MAP operations. The receiving network entity will accept an operation only if the time-stamp is within a certain time-window, the *TVP period.NE-Id* (6 octets) is used to create different *Initialization Vector*[78] (**IV**) values for different Network Elements within the same TVP period. The data element *Prop* (4 octets) is used to create different Initialization Vector (IV) values for different protected MAP messages within the same TVP period for one Network Element.

The protected payload has the following format depending on the protection mode in operation:

- Protection Mode 0: The protected payload of secured MAP messages in protection mode 0 is identical to the original MAP message payload in cleartext.

78 The Initialisation Vector is used for the initialisation of the encryption functions. For details see [3GP02b].

- Protection Mode 1: The protected payload of secured MAP messages in protection mode 1 is

  $$cleartext||f_7(Security\,header||cleartext),$$

  wheref_7 denotes a MAP integrity algorithm and cleartext is the payload of the original MAP message in cleartext. More precisely, f_7 is a 32-bit message authentication code applied to the concatenation of security header and cleartext with the use of the integrity key defined by the security association. This enables them to achieve authentication of origin and message integrity.

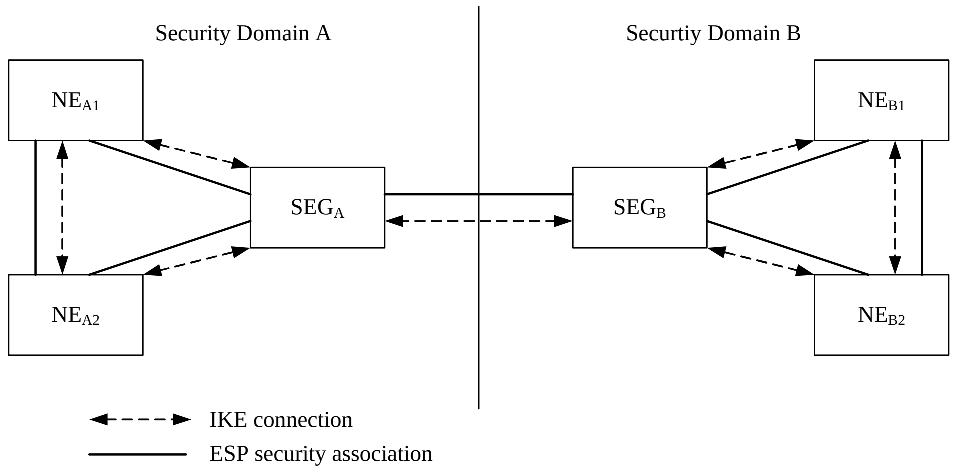
- Protection Mode 2: The protected payload of secured MAP messages in protection mode 2 is

  $$f_6(cleartext)||f_7(Security\,header||f_6(cleartext)),$$

  where f_6 is a MAP encryption algorithm and cleartext is the payload of the original MAP message in cleartext. Confidentiality is achieved by encrypting cleartext using the encryption function f_6 with the confidentiality key defined by the Security Association (SA) and the Initialization Vector (IV). Authentication of origin and integrity are achieved by applying the message authentication code function f_7 with the Integrity Key (IK) defined by the Security Association to the concatenation of security header and ciphertext. The length of the ciphertext is the same as the length of the cleartext.

In the evolution process of 2G mobile systems towards 3G mobile systems, the Internet Protocol (IP) was introduced as network layer in the GPRS backbone network and then later in the UMTS network domain. Furthermore, IP-based communication is used for both signalling and user traffic. So, in order to secure the UMTS core network domain, it is necessary to secure the IP-based communication in addition to the SS7-based one.

The IPsec protocol is used to protect the data of all IP-based protocols used in the UMTS core network domain. Using IPsec, the protection is provided at the network layer. To support the use of IPsec, the core network domain is divided logically and physically into security domains which are separated by means of *Security Gateways* denoted by **SEG**(s). SEGs are entities on the borders of the IP security domains, so that all IP traffic shall pass through them before entering or leaving the security domain. The SEGs use *Internet Key Exchange* (**IKE**)to negotiate, establish and maintain a secure *Encapsulation Security* (**ESP**) tunnel between them. The tunnel is subsequently used for forwarding IP traffic between security domain A and security domain B (see Fig. 8.3-14). For all communication within the same security domain the use ESP+IKE connections can be optionally implementet (see [3GP02a]).



ESP	- Encapsulation Security Payload
IKE	- Internet Key Exchange
NE	- Network Elements
SEG	- Security Gateways

Fig. 8.3-14: Network domain security for IP-based protocols

8.4 Security in WAP

In mid 1997 Ericsson, Motorola, Nokia and Phone.com decided to define a new protocol for mobile devices in order to offer wireless data communication services to end users, both in the form of Telecommunications-related and Internet-oriented applications. In December 1997, the four partners formed a company, WAP Forum Ltd., to control and manage the development of WAP. WAP stands for *Wireless Application Protocol* representing a protocol that uses three important network technologies: wireless data, telephony, and the Internet.

Because the WAP programming model is based on the WWW model, we first give an overview of the WWW architecture. In the WWW architecture, applications and content are presented in standard data formats, and are browsed by applications known as web browsers. A web browser is a network application, i.e. it sends requests for named data objects to a network server and the server responds with the data encoded using the standard formats [Wap01a] (Fig. 8.4-1).

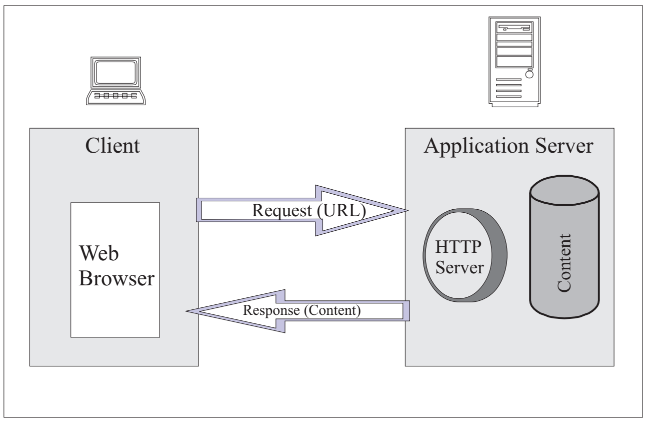


Fig. 8.4-1: WWW architecture

8.4.1 Introduction to WAP

The WAP programming model is the same as the WWW programming model with a few enhancements. Adopting the WWW programming model provides the WAP programming model with several benefits including a familiar programming model, a proven architecture, and the ability to use existing tools (e.g., web servers, XML tools, etc.). Fig. 8.4-2 shows the most significant enhancements that WAP has added. It can be seen that the WAP programming model supports the so called **Push** and the wireless telephony application (**WTA**) technologies. WTA provides the WAP client with telephony functionality such as call control, phone book and messaging. The Push technology is a new technology in transmitting information between client and server. In the normal client/server model, a client requests a service or information from a server, which then responds in transmitting information to the client. This is known as **Pull** technology. An example of this technology is browsing the World Wide Web, where a user enters an URL (the request) that is sent to a server, and the server answers by sending a Web page (the response) to the user.

The push technology is in fact based on the client/server model, but does not require an explicit request from the client before the server transmits its content. That is, in the WAP Push, informations can be transmitted to a device without an explicit user request.

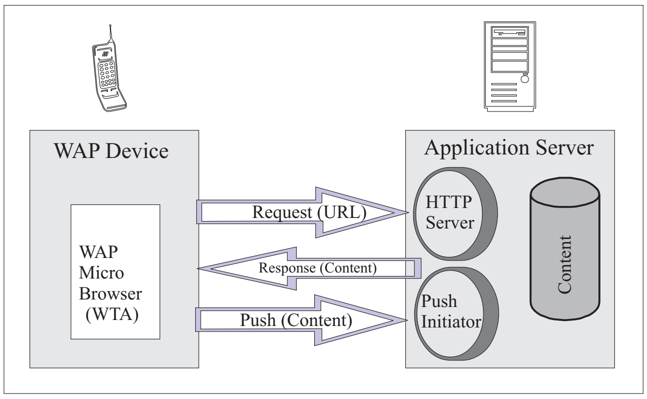


Fig. 8.4-2: WAP programming model

In order to optimize and enhance the connection between the wireless domain and the WWW, WAP utilizes the proxy technology (Fig. 8.4-3). The WAP proxy may provide a variety of functions, including [Wap01a]:

- Protocol gateway: The protocol gateway translates requests from a wireless protocol stack to the WWW protocols (HTTP and TCP/IP). The gateway also performs DNS lookups of the servers named by the client in the requested URLs.
- Content encoders and decoders: The content encoders can be used to translate WAP content into a compact format that allows better utilization of the underlying (over the air) link due to its reduced bandwidth.

- User agent profile management: User agent profiles describing client capabilities (input and output capabilities, display size, etc.) and personal preferences that are content presentation preferences.
- Caching proxy: A caching proxy can improve the performance of network utilization by maintaining a cache of frequently accessed resources (URLs).
- In addition, for a WAP proxy it is necessary to offer Push functionality.

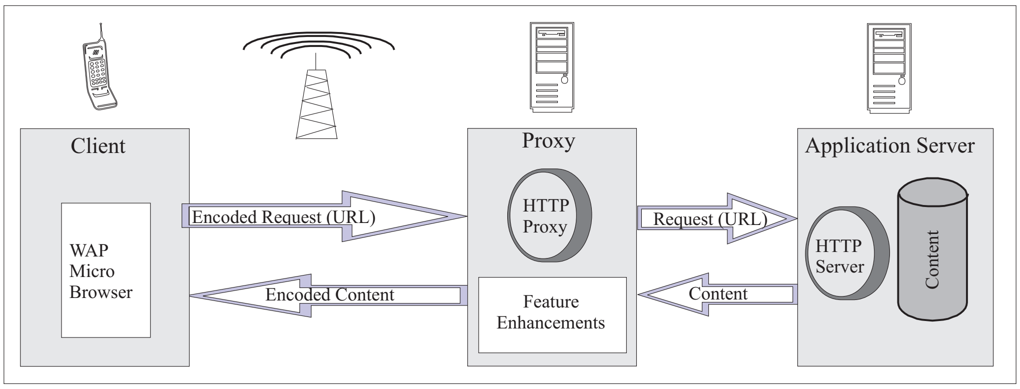


Fig. 8.4-3: WAP programming model enhanced with the proxy technology

8.4.2 The WAP Protocol Stack

The design of the WAP protocol stack takes into consideration the nature of the wireless environment and the network. Applications developed for the Internet – as the main wired network – are designed for desktop computers and medium to high bandwidth. Wireless devices present, however, a more constrained computing environment compared to desktop computers. Wireless devices tend to have less powerful CPUs, less memory (ROM and RAM), restricted power consumption, smaller displays, and different input devices (e.g. a phone keypad). Similarly, wireless data networks present a more constrained communication environment compared to wired networks. Wireless data networks tend to have less bandwidth, more latency, less connection stability, and less predictable availability.

The WAP protocol stack has a layered design (Fig. 8.4-4). Each of the layers of the architecture is accessible by the layers above, as well as by other services and applications.

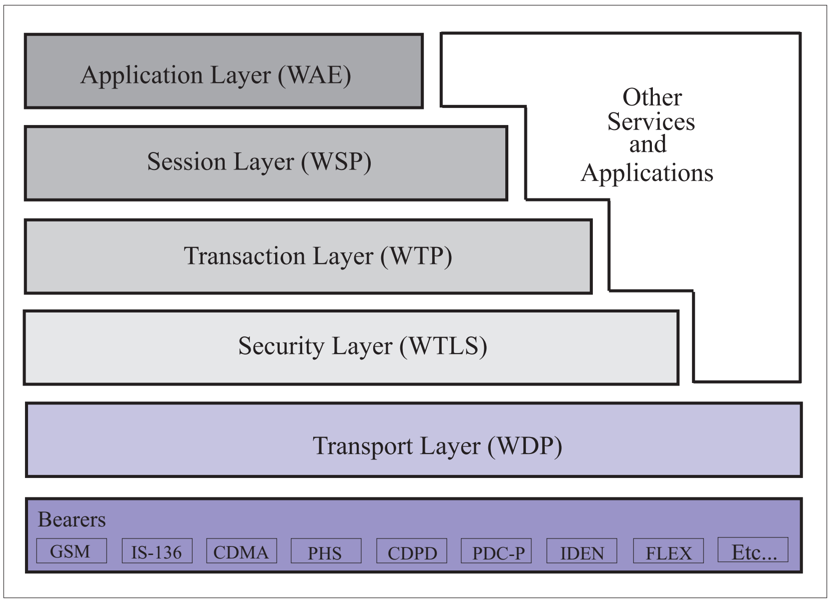


Fig. 8.4-4: The WAP protocol stack

It can be seen from Fig. 8.4-4 that the WAP protocol can operate over different bearer services. The bearer is a data transport mechanism used to carry the *Wireless Datagram Protocol* (**WDP**) between two devices. Examples of underlying bearers are GSM SMS (Short Message Service), GSM USSD (Unstructured Supplementary Service Data), GSM CSD (Circuit Switched Data), etc. WDP offers a consistent service to the upper protocols (Security, Transaction and Session) of WAP and communicates transparently over one of the available bearer services. The WDP services include application addressing by port numbers, optional segmentation and reassembly and optional error detection. The *Wireless Transaction Protocol* (**WTP**) is defined to provide the services necessary for interactive browsing (request/response) applications. The request/response cycle is referred to as a transaction. The objective of the protocol is to reliably deliver the transaction while balancing the amount of reliability required for the application with the cost of delivering the reliability [Wap01d]. The benefits of using WTP in the WAP protocol include:

- Improved reliability over datagram services. WTP relieves the upper layer from retransmissions and acknowledgements which are necessary if datagram services are used.
- Improved efficiency over connection oriented services. WTP has no explicit connection setup or termination (teardown) phases and hence no excessive overhead on the communication link.
- WTP is designed for services oriented towards transactions (request/response applications), such as browsing.

The *Wireless Session Protocol* (**WSP**) provides the upper-level application layer of WAP with a consistent interface for two session services. The first is a connection-mode service that operates above the transaction layer protocol WTP, and the second is a connectionless service that operates above a secure or non-secure datagram transport service (WDP). The Wireless session protocol currently offers services most suited for browsing applications (connection-mode service). Especially, WSP provides HTTP 1.1 functionality and incorporates new features such as long-lived sessions, capability negotiation and session suspend/resume [Wap01c].

The part of WAP used by developers to develop their wireless applications and services is the *Wireless Application Environment* (**WAE**). The WAE consists of three main parts: the *wireless markup language* (**WML**), the *wireless markup language script* (**WMLScript**), and the *wireless telephony application interface* (**WTAI**)[79].

WML is a markup language based on XML but optimized for devices with limited graphical capabilities and a small amount of screen area. WML[80] can be encoded by replacing parts of the WML (tags, attribute names and attribute values) by single byte tokens. This tokenisation improves the efficiency of transmission. WMLScript is a scripting language (like JavaScript) based on a subset of the ECMAScript (European Computer Manufacturer Association) WWW scripting language. WMLScript can be used together with WML to provide intelligence to the clients. For example, WMLScript provides access to the WAP device and its peripherals. One important feature of WMLScript is that it can be compiled to a compressed bytecode and hence directly interpreted by a virtual machine[81]. Transmitting these compiled bytecode results in a reduction in size and also in the amount of parsing/processing required on the WAP enabled device. Note that the encoding/decoding of WML/WMLScript instructions into instructions understood by the Internet is done in the WAP gateway as mentioned in previous sections.

8.4.3 Wireless Transport Layer Security (WTLS)

Security in WAP is defined by the *Wireless Transport Layer Security* (**WTLS**) protocol. In the WAP stack, the WTLS protocol operates above the transport layer protocol. The WTLS protocol is a derivate of the TLS/SSL[82] protocol but is optimized for low-bandwidth bearer networks with relatively high latency. The use of WTLS is not mandatory for each application, but only for those requiring a certain level of security. The cryptographic goals reached by using WTLS are confidentiality, data integrity, and authentication between two communicating applications. WTLS does not, however, provide non repudiation services.

The WTLS protocol is based on a client-server architecture. In order to establish a secure session between client (WAP device) and server (Web server), the two communicating parties first have to negotiate security parameters which will be used to secure the session. This is done by performing the **handshake protocol** which is a

79 See [Heijden00] for more details.

80 WML (Version 1).

81 This is not the case with WML which is only a compact representation of the content.

82 A detailed description of the TLS/SSL protocols is given in the course Network Security in chapter 3.

subprotocol of the WTLS protocol. During the handshake protocol, the communicating parties agree on a protocol version, select cryptographic algorithms, optionally authenticate each other, and use public key encryption techniques to generate a shared secret. The message flow of the handshake protocol is illustrated in Fig. 8.4-5.

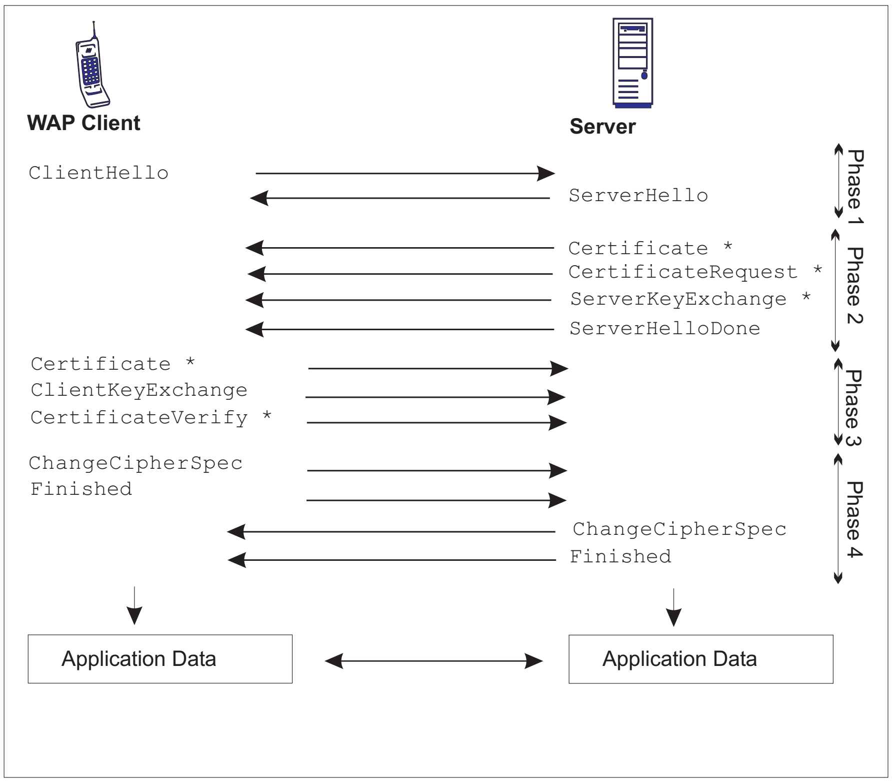


Fig. 8.4-5: Message flow of the handshake protocol

The Handshake Protocol consists of a series of messages exchanged by client and server and can be viewed as having four phases. In the **first phase security capabilities are established**. The phase is initiated by the client and is used to initiate a logical connection by exchanging hello messages between the client and the server.

When a client first connects to a server it is required to send the `ClientHello` as its first message (see Fig. 8.4-5). The client can also send a `ClientHello` in response to a `HelloRequest` message that comes from the server or on its own initiative in order to renegotiate the security parameters in an existing connection. The `ClientHello` message contains, among other things, the following parameters: `Version` (the latest WTLS version understood by the client), `Session ID` (session identifier used to identify a secure session between client and server), `Random Structure` (12 random bytes, `HelloClient.random`, generated by a secure random number generator and used later in the protocol), `Cipher Suite` (The `Cipher Suite` defines a bulk encryption algorithm (including secret key length) and a MAC algorithm. Examples of a bulk encryption algorithm and a MAC algorithm are DES algorithm operating in CBC mode with 40 bits key length and MD5 MAC-algorithm with 128 bits key length, respectively.), and `Key`

`Exchange Suite` (a list of key exchange algorithms supported by the client, such as RSA or anonymous Diffie-Hellman key exchange algorithm).

The server processes the `ClientHello` message and responds with either a `handshake failure` (fatal error will occur and the connection will fail) or `ServerHello` message. Generally, the `ServerHello` message contains the same parameters as the `ClientHello` message, such as `Version`, `Random`, `Session ID`, and `Cipher Suite`. The `Version` field contains the lower of the WTLS version suggested by the client and the highest supported by the server. The `Random` field, denoted `Helloserver.random`, is generated by the server. If the server is willing to establish the new connection using the specified secure session[83], it will respond with the same value of session identifier supplied by the client. The `Cipher Suite` parameter contains a single cipher suite selected by the server from the list of cipher suites proposed by the client (if no acceptable choices are presented, the server returns a `handshake failure` alert and closes the connection). The `ServerHello` message also includes a field containing the number of the key exchange suite selected by the server from the list in the `ClientHello` message.

Now after the end of hello messages, security enhancement capabilities are established between client and server, i.e. the protocol version, session ID, cipher suite, key exchange suite, etc.

The **second phase** of the handshake protocol is the **Authentication and Key Exchange**. If the server is to be authenticated (which is generally the case), it sends its certificate with the `Certificate` message immediately following the `ServerHello` message. The certificate type must be appropriate for the selected key exchange suite's algorithm. It can be a X.509.v3 certificate, a WTLS certificate which is optimized for size, or some other. If the server has no certificate, or its certificate is used for signing only, the server sends the **`ServerKeyExchange`** message. With this message, the server informs the client about cryptographic parameters which enable the client to exchange a premaster secret , i.e., the message contains the parameters used for RSA key exchange (RSA modulus and the public exponent of the server's temporary RSA key), the parameters used for Diffie-Hellman key exchange (prime modulo p, generator g, and the server's public value $g^x \bmod p$).

When the server is authenticated, it may request a certificate from the client if that is appropriate to the cipher suite selected, and send the `CertificateRequest` message. This message consists of a list of the types and names of acceptable certification authorities. The final message in phase 2, and the one that is always required, is the `ServerHelloDone` message sent by the server. It indicates that the hello-message phase of the handshake is complete. The server will then wait for a client response.

83 Which is a secure session established previously between the same client and server.

The **third phase** of the handshake is the **client authentication and key exchange**. Upon receipt of`ServerHelloDone` message, the client should verify that the server provided a valid certificate if required and check that the `ServerHello` parameters are acceptable. If all is satisfactory, the client sends one or more messages back to the server. If the server has sent a `CertificateRequest` message, the first message the client sends is its certificate in the `Certificate` message (or a no certificate message, if no suitable certificate is available).

Next, the **`Client Key Exchange`** message is sent. The content of this message depends on the type of key exchange selected during the exchange of the `ClientHello` and the `ServerHello` messages. With sending this message, a premaster secret is set. For example, if RSA is used for key agreement and authentication, the client generates a 20-byte premaster secret, encrypts it with the public key from the server's certificate, and sends the result in an `encrypted secret` message. This random value (premaster-secret) is generated by the client and is used to generate the master secret. Finally, in this phase, the client may send a `CertificateVerify` message to provide explicit verification of the client certificate (if the client has sent a certificate). This message signs a hash code based on some messages of the handshake. Note that the hash algorithm being used is the one agreed during the handshake.

The **fourth phase**, called the **finish phase** completes the setting up of a secure connection. The client sends a `ChangeCipherSpec` message and copies the pending CipherSpec into the current CipherSpec. Note that this message is not considered part of the Handshake Protocol but is sent using the ChangeCipherSpec Protocol, which is a subprotocol of the WTLS protocol and which notifies the other communicating party that subsequent messages (called records in the context of the WTLS protocol) will be protected under the newly negotiated CipherSpec and keys. The client then immediately sends the `Finished` message to verify that the key exchange and authentication process were successful. This is the first message protected with the just-negotiated algorithms and keys. This message includes hash values of all handshake messages starting at `ClientHello` up to, but not including, the `Finished` message.

In response to these two messages, the server sends its own `ChangeCipherSpec` message, transfers the pending to the current CipherSpec, and then sends its own `Finished` message under the new CipherSpec. Recipients of the `Finished` message must verify that the contents are correct. No acknowledgement of the `Finished` message is required.

At this point, the handshake is complete and the client and server may begin to exchange application layer data using the secure connection.

When the client and server decide to **resume a previous session** instead of negotiating new security parameters, the abbreviated handshake, which is a variant of the (full) handshake protocol, is to be used (Fig. 8.4-6).

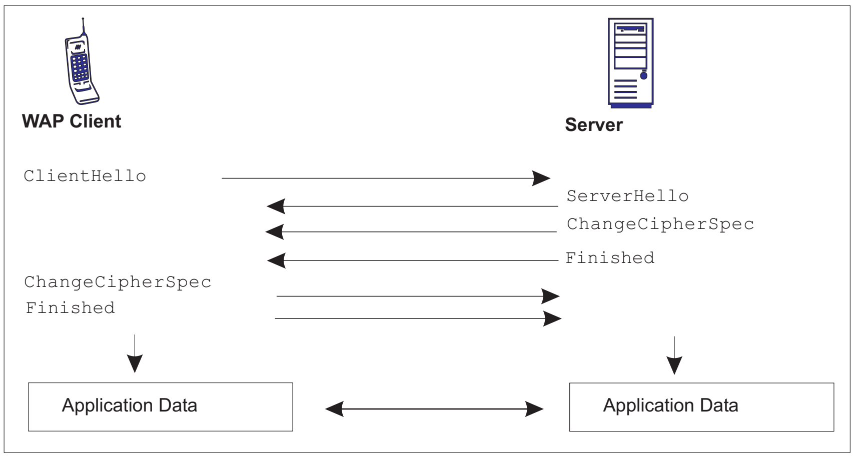


Fig. 8.4-6: Message flow of the abbreviated handshake

8.4.4 End-to-end Security in a WAP Connection

An important question when using WTLS is whether end-to-end security or security through a proxy is provided. As mentioned previously, the WAP technology utilizes the proxy technology (Fig. 8.4-3) in order to optimize and enhance connection between the wireless domain and the WWW domain. The level of security between WAP client and server will then depend on the position of the gateway in the system architecture.

WAP gateway hosted by the content provider

In this case (Fig. 8.4-7) the content provider network is assumed to be trusted. The WAP client establishes a WTLS secure connection to the WAP gateway, which makes the necessary translation of the WAP stack into the Internet stack (this is done because of the incompatibility between the WTLS protocol and the TLS/SSL protocol), and forwards the WAP client request to the web server. The latter processes the WAP client request and responds with a WAP client response that is forwarded to the WAP gateway. The WAP gateway translates it from the Internet stack into the WAP stack. After translation, the WAP client response is directed over the air to the WAP client through the WTLS secure connection.

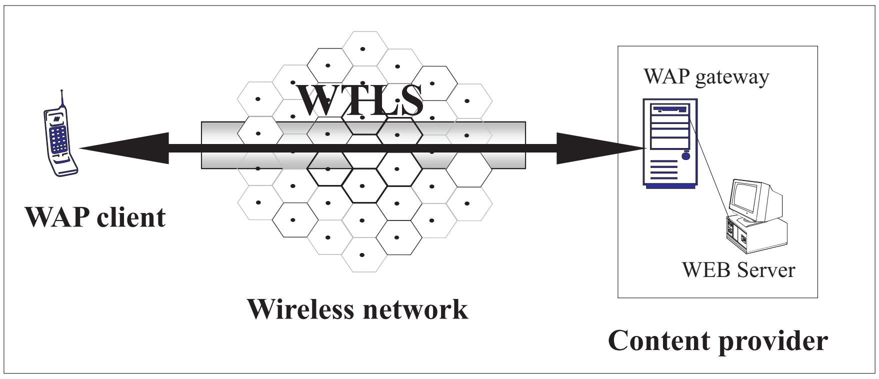


Fig. 8.4-7: WAP gateway hosted by the content provider

In this scenario, end-to-end security is reached since all communication within the content provider network is assumed to be protected and secured. The client and the WAP gateway can be authenticated by the use of the certificates negotiated during the handshake protocol. In addition, the communication between WAP client and gateway is encrypted and integrity protected by using the encryption algorithm and the MAC algorithm agreed to in the handshake protocol.

This model provides end-to-end security, but in practice it cannot be widely used since it is not always cost-effective that the content provider is in possession of its own WAP gateway[84].

WAP gateway hosted by the network provider

When the WAP gateway is hosted by the network provider, two independent security zones are built: WAP client-WAP gateway security and WAP gateway-Web server security (Fig. 8.4-8).

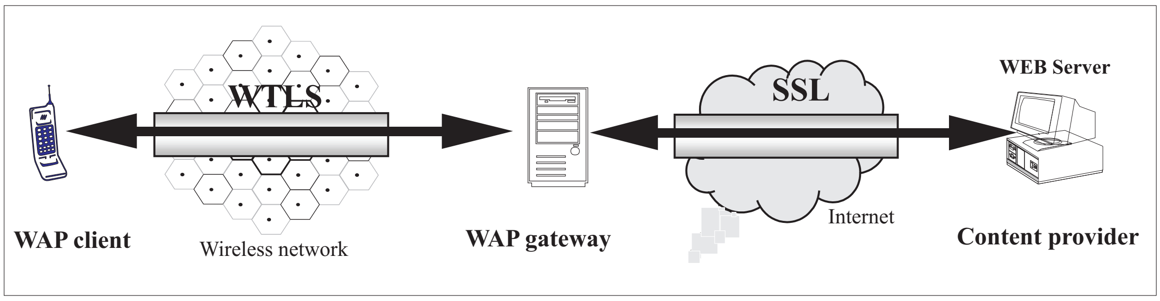


Fig. 8.4-8: WAP gateway hosted by the network provider, no end-to-end security

The communication between WAP client and WAP gateway is secured through the WTLS tunnel. In the WAP gateway, the WTLS-secured content must be decrypted by the WAP client and then encrypted using TLS/SSL before passing it to the Web server. Thereafter, the WAP gateway establishes a secure TLS/SSL tunnel to the web server and forwards the traffic through this tunnel. Hence, the content can be accessed in clear mode by the network provider. It is obvious that no end-to-end security is provided in this scenario. The WAP client and the Web server must have considerable trust to the WAP gateway, which is normally not the case. The WAP client has no proof whether an SSL connection has really been established between the WAP gateway and the Web server. Similarly, the Web server cannot confirm whether a secure WTLS has been built between the WAP client and the Web server. The WAP client can be authenticated to the Web server at the application layer by using password authentication. There is, however, no direct way to authenticate the Web server to the WAP client. The Web server should authenticate itself to the WAP

84 One solution to this problem that provides end-to-end security of WAP applications with minimal costs is proposed in [Tzvetkov].

gateway using its certificate and the user of the WAP client should trust the WAP gateway that the authentication is correct.

8.4.5 Other Security Components of WAP

Other security components of WAP are the *WMLScript Crypto Library*[Wap01e] and the *Wireless Identity Module* (*WIM*) [Wap01b]. The WMLScript Crypto Library defines a mechanism to see a text string and to sign it digitally. This complements the security goals of the WAP technology to the non repudiation issue in addition to privacy, data integrity, and authentication issues.

Many kinds of applications, e.g. electronic commerce, require the ability to provide persistent proof that someone has authorized a transaction. Although WTLS provides transient client authentication for the duration of a WTLS connection, it does not provide persistent authentication for transactions that may occur during that connection. One way to provide such authentication is to associate a digital signature with data generated as the result of a transaction, such as a purchase order or other financial document [Wap01e].

The `Crypto.signText` is a WMLScript function that supports persistent authentication for application. A call to the `signText` method first displays the exact text to be signed. The user is allowed to confirm this text or not. After confirmation, the data will be signed and both the signature and the data will be sent across the network to the appropriate server. The latter can extract the digital signature and validate it.

The signature key is stored in the WIM. The WIM is a tamper-resistant device with computation capability having an interface to the WAP client. In fact, the proper digital signature computation is performed in the WIM (using signature key) before being displayed by the WAP client. The WIM has other functionalities in addition to storing private keys and digital signature computation. It can be used for WTLS operations, like performing cryptographic operations during the handshake, especially those that are used for client authentication. For example, during an RSA handshake, the WIM performs the signature operation proving the client’s identity with the use of the client public key or certificate stored in it. The WIM may also store needed certificates: CA and user certificates[85].

8.5 Security in Bluetooth

Bluetooth[86] is an open industry standard for short range, wireless connectivity between common devices, e.g. connection of wireless PC keyboard, wireless mouse,

85 See [Wap01b] for more details about the WIM functionality and capability.

86 The standard is named after Harald Blaatand "Bluetooth" II, king of Denmark 940-981A.D. A runic stone has been erected in his capitol city Jelling (Jutland) that depicts the chivalry of Harald and the "runes" say: Harald christenized the Danes; Harald controlled Denmark and Norway; Harald thinks notebooks and cellular phones should seamlessly communicate.

and wireless mobile telephone to a PC. It is intended as a replacement of cable connections in a range of up to 100 m. Bluetooth devices are categorized into three different classes dependent on their transmission power. Class 3 devices use 1 mW power and have a transmission range of 0.1 to 10 meters. Class 2 devices use 1 to 2.5 mW power and have a range of 10 meters. Class 1 device use up to 100 mW power and have a range up to 100 meters. The Bluetooth system operates in the 2.4 GHz ISM (Industrial Scientific Medicine) band.

8.5.1 Introduction to the Bluetooth Technology

A Bluetooth system provides a point-to-point connection between two Bluetooth devices, or a point-to-multipoint connection. In a point-to-multipoint connection, the channel is shared among several Bluetooth devices. A **piconet**[87] consists of two or more Bluetooth devices sharing the same channel, where one component acts as the master of the piconet and the other component(s) (up to seven) act(s) as slave(s). All communication is between the master and the slave. Slaves cannot address each other directly. It is, however, possible for devices to dynamically switch roles: for a slave to become a master and vice-versa [Dent]. Multiple piconets (up to ten) with overlapping coverage areas form a scatternet. Each piconet can only have one single master. However, slaves can participate in different piconets. In addition, a master in one piconet can be a slave in another piconet.

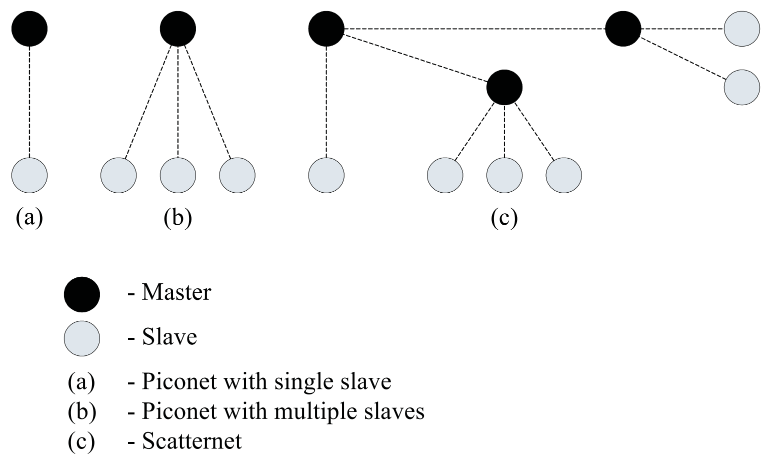

(a)
(b)
(c)
● - Master
○ - Slave
(a) - Piconet with single slave
(b) - Piconet with multiple slaves
(c) - Scatternet

Fig. 8.5-1: Piconet and scatternet.

Between master and slave(s), different types of links can be established. Two link types have been defined:

87 It is ”pico” because of the short range of the Bluetooth wireless communications.

- Synchronous Connection Oriented (SCO) link: The SCO link is a point-to-point link between a master and a single slave in a piconet. It can be considered as a circuit-switched connection. The master can support up to three SCO links to the same slave or to different slaves. A slave can establish up to three SCO links to the same master, or two SCO links if the links originate from different masters.

- Asynchronous ConnectionLess (ACL) link: The ACL link provides a packet-switched connection between the master and all active slaves participating in the piconet. Between a master and a slave only a single ACL link can exist.

How is the channel of a piconet established and how can Bluetooth devices be added to and released from the piconet? Several states of operation of the Bluetooth device are defined to support these functions. The **inquiry** state allows a device to discover other Bluetooth devices which are in its range. The **inquiry scan** state allows a device to be discovered. A device that allows itself to be discovered is in **inquiry scan** and is ready to respond to inquiry messages. Once the devices are discovered by each other, a connection can be established using the paging procedure. The **page scan** state enables a connection to be established. The **page** state is used by the master[88] (source) to activate and connect to a slave (destination) which periodically ”wakes up” in the page scan state. To ensure that connected devices do not remain active all the time, power saving modes have been defined:

- **HOLD** mode: In this mode, a slave temporarily stops transmission. The HOLD mode is used to let the master establish a link to another device. With the hold mode, capacity can be made available to the slave to do other things like scanning, paging, inquiring, or attending another piconet.

- **SNIFF** mode: In this mode, the duty cycle of the slave’s listen activity can be reduced but the slave remains an active device of the piconet. The master can only transmit after the "SNIFF" time.

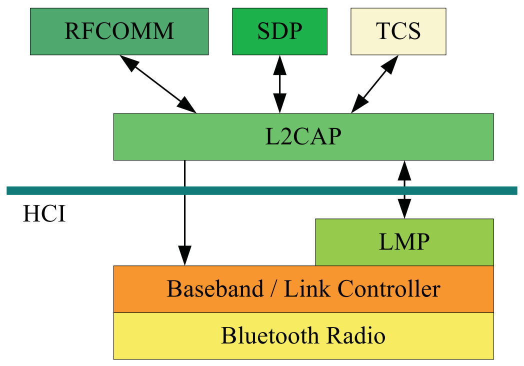
- **PARK** mode: When a slave does not need to participate in the piconet channel but wants to remain synchronized to the channel, it can enter the PARK mode. The PARK mode is a low power mode with very little activity in the slave. The PARK mode is used to connect more than seven slaves to a single master, where maximal seven of them are allowed to be active.

8.5.2 Bluetooth Protocol Stack

A Bluetooth protocol stack (see Fig. 8.5-2) has been specified[89], so that electronic equipment can be connected to a Bluetooth device.

88 The Bluetooth device that initiates the connection plays the role of a master.

89 The Bluetooth specification is developed by the Bluetooth Special Interest Group SIG, which is an association of telecommunication, computing, and network companies.



HCI	- Host Controller Interface
L2CAP	- Logic Link Control and Adaption Protocol
LMP	- Link Manager Protocol
RFCOMM	- Radio Frequency Communication Protocol
SDP	- Service Discovery Protocol
TCS	- Telephony Control Protocol

Fig. 8.5-2: The Bluetooth stack

The RF (Radio Frequency) layer is responsible for modulation/demodulation of signals to be transmitted/received over the air by the Bluetooth device. *Frequency Hopping* is used to minimize the effect of interference and fading. The link controller carries out the baseband protocols. The function of the baseband protocol includes the handling of SCO and ACL packets and controlling the physical links via the radio (error correction). The LMP protocol provides link set up (SCO/ACL links), link configuration, and security function. The link configuration tasks include quality of service, power control, and handling the mode of operation of the device (HOLD mode, SNIFF mode, etc). The security function deals with the authentication of the Bluetooth devices to be linked, with the encryption of the traffic in a piconet, and with the management of the cryptographic keys. The Host Controller Interface is used to isolate the Bluetooth baseband and link manager from a transport protocol such as USB or RS-232. This allows a standard host processor to connect to Bluetooth hardware[90]. An HCI driver on the host is needed to interface Bluetooth applications with the transport protocol.

The L2CAP protocol supports multiplexing, segmentation and reassembly of ACL packets, and the conveying of quality of service information. With multiplexing, the L2CAP protocol allows multiple applications to use a link between two devices simultaneously. In the segmentation process, packets received by applications are reduced in a size accepted by the baseband. On the reverse way, baseband packets are reassembled before being forwarded to the application. The L2CAP protocol allows Bluetooth devices to exchange informations about the quality of service expected in the connection in use. In addition, it supports group management mapping upper protocol groups to Bluetooth piconets. Several Bluetooth protocols interface to the L2CAP link layer. SDP provides service discovery specific to the Bluetooth environment (example of services are printing, paging, FAXing, and information access services such as teleconferencing, and eCommerce facili-

90 The Bluetooth hardware integrates the RF, the baseband, and the link manager.

ties) and determines the characteristics of the service discovered. The RFCOMM protocol allows applications developed for serial ports to run (to be emulated) on Bluetooth platforms without modification. The Telephony Control protocol Specification (TCS) is responsible for:

- Call Control (CC): Signalling for the establishment and release of speech and data calls between Bluetooth devices.
- Group Management: Signalling to ease the handling of groups of Bluetooth devices.
- ConnectionLess TCS (CL): TCS CL is used to exchange signalling information without establishing a TCS call. For example, when a home base station needs to alert all phones in range for an incoming call.

Fig. 8.5-2 illustrates a relatively abstract form of the Bluetooth stack. In addition, the SIG has specified profile stack specifications for application protocols like FTP, and LAN access protocol.

8.5.3 Security Features and Keys

The security provided by Bluetooth is based on authentication of the devices willing to establish a connection and the encryption of data passing through this connection. In fact, the security features are specified at the link level. During the connection establishment, Bluetooth devices can negotiate the security level of the communication depending on which security mode the Bluetooth device is in. Three security modes have been specified:

- Security mode 1 (non-secure): They shall never initiate any security procedure.
- Security mode 2 (service level enforced security): The channel or service using L2CAP connection decides whether or not security is required. So until an L2CAP channel has been established, a device will not initiate any security procedures. A Bluetooth device in security mode 2 should classify the security requirements of its services using at least the following attributes: Authorization required, authentication required, encryption required [Loehlein02].
- Security mode 3 (link level enforced security): In this mode, the authentication of the devices is part of the connection establishment procedure. Encryption of the communication is optional.

The key used for authentication is named link key. Moreover, it is used as one of the parameters for generating the encryption key. In fact, the exchange of the keys takes place during an initialization phase which consists of:

- Generation of an initialization key,
- Generation of a link key,
- Link key exchange,
- Authentication, and

- Generation of an encryption key in each Bluetooth device (optional).

Four types of link keys are specified:

- The initialization key K_{init}: It is used as link key during the initialization process when no combination or unit keys (which are explained later) have been defined and exchanged yet or when a link key has been lost. K_{init} is derived by the E_{22} algorithm from a PIN code, the length of the PIN (in octets), and a random number IN_RAND (128 bit) (Fig. 8.5-3). The E_{22} algorithm is based on the block cipher SAFER+[91]. The PIN code can be entered manually by the user or can be generated automatically by a specific application. The PIN code can vary from 8 to 128 bits. Once k_{init} has been derived, the PIN code will be augmented with BD_ADDR. BD_ADDR is the 48-bit public address of the Bluetooth device.

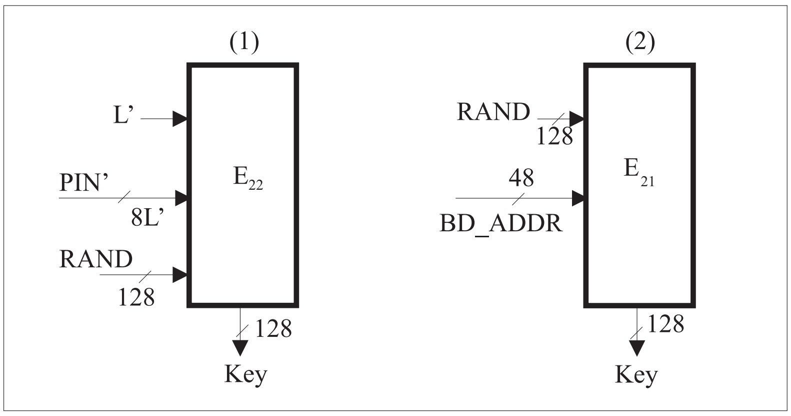


Fig. 8.5-3: Key generation for k_{master} and K_{init} in (1) and for the unit key and combination key in (2)

- The unit key k_A: This key is generated in a single device A. It is generated once the Bluetooth device is in operation for the first time. The unit key is used for authentication by Bluetooth devices with less memory to store keys. The unit key is computed by the E_{21} algorithm (based on the block cipher SAFER+, too) from a 128-bit random number and the BD_ADDR address (see Fig. 8.5-3). The unit key is exchanged by XORing it with the initialization key K_{init}.

The combination key k_{AB}: This key is generated by the use of information in both devices A and B, which are 128 bits random numbers LK_RAND_A and LK_RAND_B generated in the devices A and B, respectively. It is suitable for applications which require a higher level of security. However, these applicati-ons must run in an environment with more memory since for every connection with a new device a new combination key has to be stored. The combination key k_{AB} is computed by using the algorithm E_{21}, the random numbers (LK_RAND_A, LK_RAND_B) and the own address (BD_ADDR_A, BD_ADDR_A) as input. First, the numbers LK_K_A, LK_K_B are created in the corresponding device as follows:

91 SAFER+ is one of the fifteen block cipher finalists that were selected as AES candidates.

$LK\_K_A = E_{21}(LK\_RAND_A, BD\_ADDR_A)$ and

$LK\_K_B = E_{21}(LK\_RAND_B, BD\_ADDR_B)$.

Then, the two random numbers LK_K_A and LK_K_B are exchanged securely by XORing them with the current link key, k (in the initialization phase $k = K_{init}$). Now each device can reveal the random number of the other device by XORing the received message with the current link key k (Fig. 8.5-5). Finally, the combination key k_{AB} is calculated by XORing the two numbers LK_K_A and LK_K_B. The old link key k shall be discarded after generation of the combination key k_{AB} (the new link key).

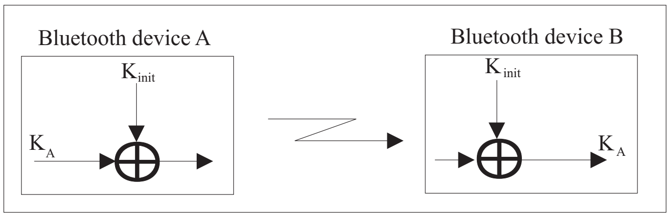


Fig. 8.5-4: Exchange of the unit key k_A, where Bluetooth device A is the device with less memory

- The master key k_{master}: It is used for point-to-multipoint communication, where it is not useful to generate cryptographic keys for each pair of devices. The master key k_{master} is generated in the same way as K_{init} (Fig. 8.5-4) and is used temporary only in one session.

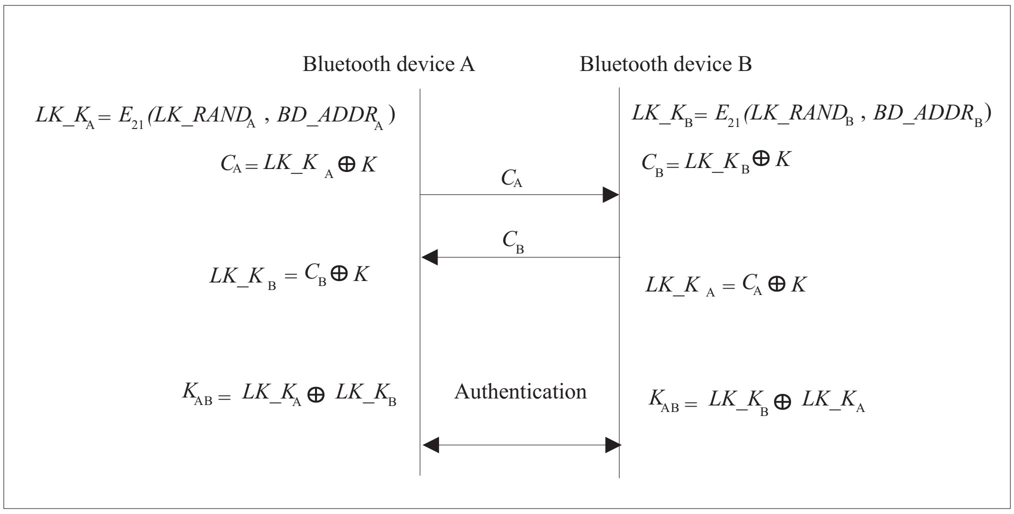


Fig. 8.5-5: Generation and exchange of the combination key

8.5.4 Authentication

The authentication of Bluetooth devices is based on a challenge-response scheme. Fig. 8.5-6 shows how a Bluetooth device A can authenticate the device B. For this purpose, the claimant (device B) must prove its knowledge of a shared secret, denoted here as AU_RAND_A. First, the verifier computes an expected response $SRES'$ from the inputs AU_RAND_A, BD_ADDR_B, and the link key. AU_RAND_A is a 128 bit random number generated by A. BD_ADDR_B is the device address of the claimant which can be obtained by an inquiry procedure. E_1 is a MAC algorithm based on SAFER+. The output of E_1 is a 128 bit hash value, where the first 32 bits constitute the response $SRES'$ and the rest 96 bits form a number called ACO (Authenticated Ciphering Offset). ACO is used in the generation of the encryption key. Thereafter, the verifier sends the challenge AU_RAND_A to the claimant. After receiving the challenge, the claimant computes the response $SRES$. $SRES$ are the first 32 bits taken from the hash value generated by the E_1 algorithm and the inputs AU_RAND_A, BD_ADDR_B, and the link key. $SRES$ is then sent to the verifier which checks whether $SRES$ and $SRES'$ are equal. If they are equal,device B is authenticated by device A.

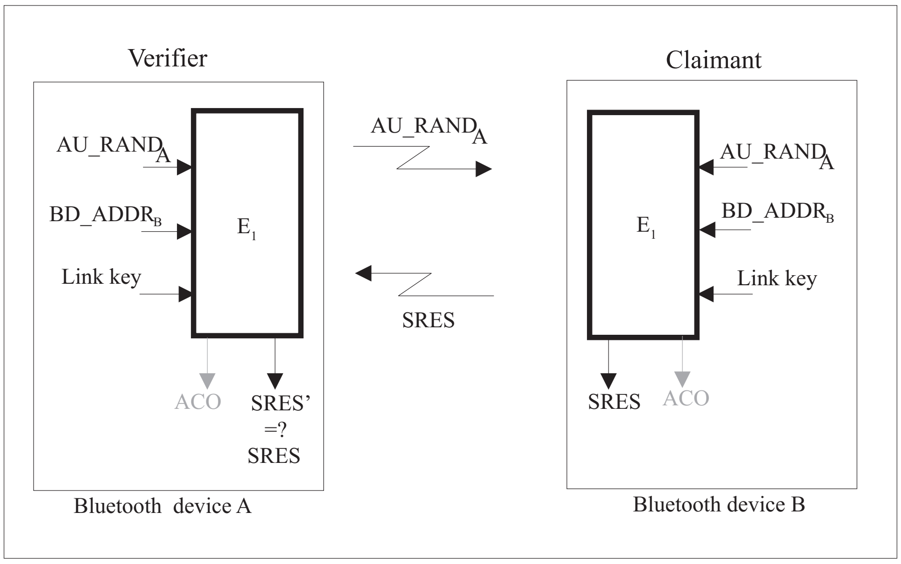


Fig. 8.5-6: Authentication in Bluetooth

Note that the verifier must not necessarily be a master. On the other hand, if mutual authentication is to be used, device B can challenge device A in the same way, using a challenge-response scheme with another 128 bit random number AU_RAND_B generated by him.

8.5.5 Encryption

The encryption in Bluetooth protects the payload of the packets transmitted between Bluetooth devices against eavesdropping. The key used for encryption is denoted as K_c and is generated by the MAC algorithm E_3[92] with the inputs EN_RAND, COF (Ciphering Offset), and the link key (Fig. 8.5-7). EN_RAND is a public 128-bit random number issued by the master. COF is a 96-bit number defined as follows:

$$COF = \begin{cases} BD\_ADDR || BD\_ADDR & \text{if link key is a master key} \\ ACO & \text{otherwise.} \end{cases}$$

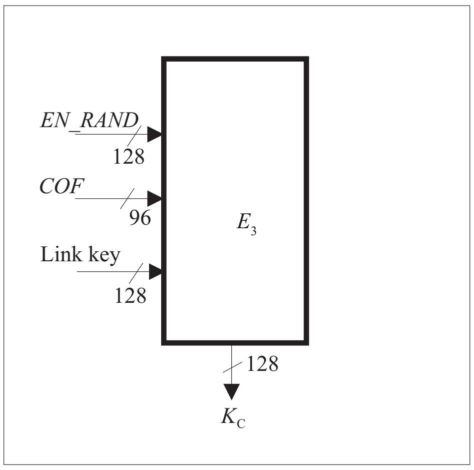


Fig. 8.5-7: Encryption key generation

The encryption of the payload is performed with the stream cipher called E_0 that is resynchronized for every payload. It consists of bitwise XORing of data stream and keystream bits generated by E_0. The cipher algorithm E_0 uses the Bluetooth master address, 26 bits of the master real-time clock (CLK_{26-1}) and the encryption key as input K_c, (see Fig. 8.5-8, where the Bluetooth device A is the master).

92 E_3 is based on SAFER+, too.

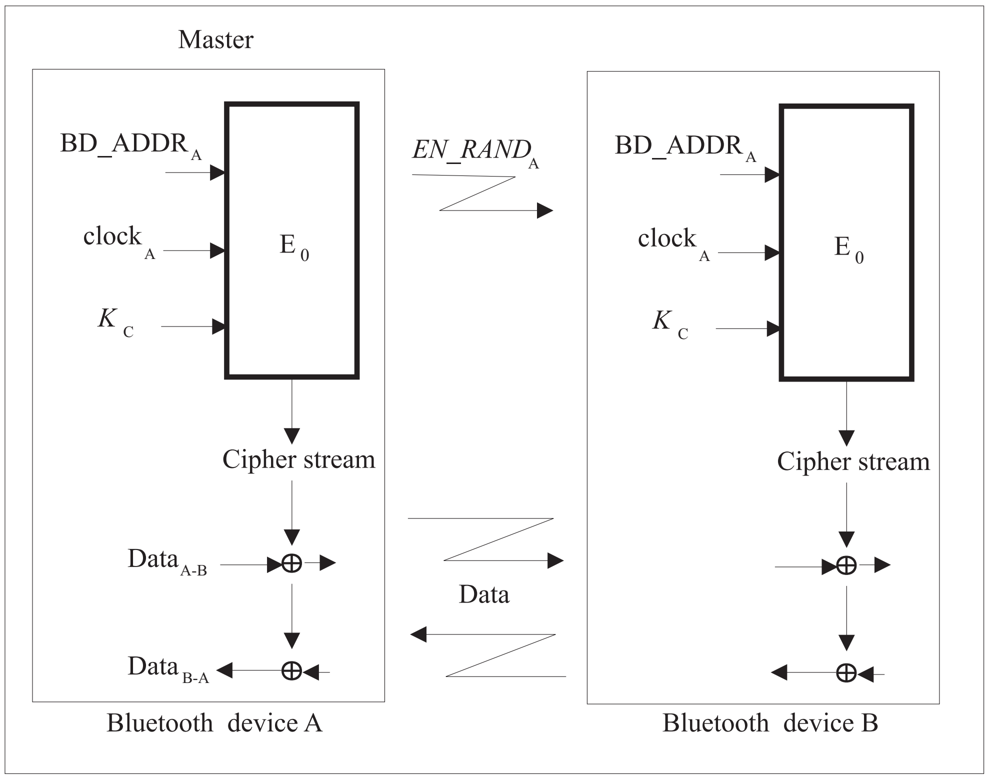


Fig. 8.5-8: The Bluetooth encryption procedure.

Within the algorithm E_0, the encryption key K_c (128 bit) is set to another key k'_c, which is the same as K_c but with modified bitlength L, ($8 \leq L \leq 128$). The size of k'_c is negotiated between the communicating devices before the encryption takes place. The realtime clock is incremented for each slot. The algorithm E_0 is reinitialized at the start of each new packet (i.e. for master-to-slave as well as for slave-to-master transmission). By using CLK_{26-1} at least one bit is changed between two transmissions. Thus, a new keystream is generated after each reinitialization. For packets covering more than a single slot, the Bluetooth clock as found in the first slot is used for the entire packet [Http].

8.5.6 Security Weaknesses

The Bluetooth security is not optimal and some security weaknesses were addressed in recent research publications (e.g. [Loehlein02], [Fox02a]). All authentication procedures and key generations depend on the secret PIN. If the PIN is not changed for a long time and if it is not of considerable length (at least 80 bit), then it will be vulnerable to discovery. In this case the whole security system will fail.

As already mentioned, the unit key, that is an entity identifying the device, can be used as a link key. In other words, it can be in the hand of other devices. Once in possession of a link key, an attacker can launch impersonation attacks. It is preferable to change the link key for every session or to use a combination key as link key.

The Bluetooth device address BD_ADDR is publicly known, thus it can be easily spoofed and thereafter misused. Once BD_ADDR is associated with a person, an attacker can determine the location and movements of victim devices (location tracking) by maintaining geographically distributed devices that continuously inquire all devices within reach and recording the identities given in the response [Loehlein02]. A solution is to randomly change BD_ADDR for each connection.

SAFER+ was chosen in the Bluetooth specification as the base algorithm for authentication and key generation. SAFER+ is one of the fifteen block cipher finalists in the AES selection process[93], but it did not reach a selection to the five finalists. So, better algorithms could be used.

In addition to these security weaknesses, the Bluetooth specification has not addressed data integrity at the link level. More security weaknesses can be found in [Loehlein02].

8.6 Security in IEEE 802.11

Institute of Electrical and Electronics Engineers

The first version of the ***I**nstitute of **E**lectrical and **E**lectronics **E**ngineers* (IEEE) standard for wireless networks was developed in 1997 in the working group 802.11. In this version the physical and the data link layer were specified. Additional security features were developed in the following years and will be discussed in this chapter.

8.6.1 Wired Equivalent Privacy (WEP)

In the year 1999 two additional specifications followed: IEEE 802.11a and IEEE 802.11b. Beside thw specifications in the physical network layer, two different security aspects were treated: authentication and privacy (encryption). The next two subsections describe these security aspects and after this Section 8.6.1 will close with a discussion about several security flaws of the protocol.

8.6.1.1 Authentication

Frame Types

In the IEEE 802.11 protocol there are three different types of frames defined: control, management and data frames. For the authentication process management frames are used in both available authentication procedures: *Open System Authentication* and *Shared Key Authentication*. The structure of management frames (or the frame format of IEEE 802.11 at all) will not be discussed in this course, but we point out the neccessary fields for the authentication process defined in subtype `1011` (authentication) of a management frame. A detailed discussion about IEEE 802.11 can be found in [Re04].

93 See the section Advanced Encryption Standard in Chapter 4 in the course Foundations of Cryptology.

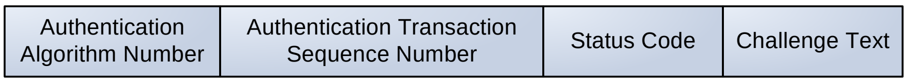

Authentication Algorithm Number
Authentication Transaction Sequence Number
Status Code
Challenge Text

Fig. 8.6-1: Management frame: Authentication message format

Authentication Algorithm Number defines the authentication algorithm in a 16 bit field. The value of `0` indicates the *Open System Authentication*, `1` shows *Shared Key Authentication*.

Authentication Transaction Sequence Number defines the actual status of an authentication in progress. The 16 bit counter starts in the first message with 1 and is incremented in each following step.

Status Code notifies the success or failure of a previos operation. In the 16 bit field there are different status codes defined, e.g. *success* (`0`), *authentication failure because of errors in the challenge-text* (`15`) or *not supported authentication algorithm* (`13`).

Challenge Text is a subtype of the field *information element* with a flexible length from 3 to 255 bytes. The information element has a simple frame structure with three subelements. The first one is the *information element id* which indicates the information type in a 1 byte field. A challenge text has the *information element id* of `16` and is followed by a 1 byte field that indicates the *length* of the *challenge text* itself. The last element in the frame will be the *challenge text* with a length from 1 to 253 bytes.

Open System Authentication

The *Open System Authentication* is the only scheme which is required by the IEEE 802.11 specification. A mobile station is authenticated if it simply requests authentication with it's MAC address. Obviously this scheme does not provide any kind of authentication.

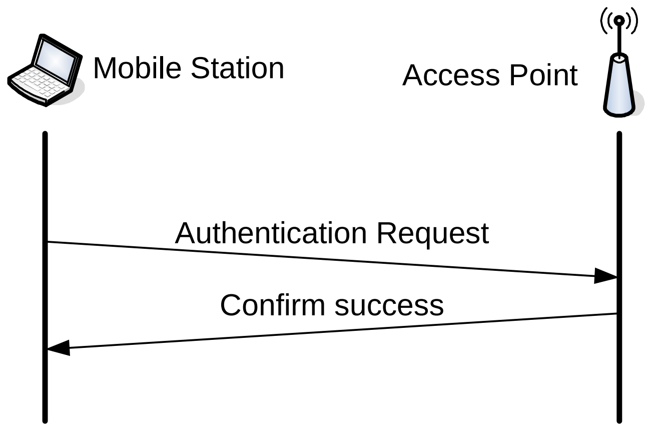


Fig. 8.6-2: Open System Authentication

Shared Key Authentication

Shared Key Authentication is a challenge-response scheme and is based on the knowledge of a shared secret. As shown in Fig. 8.6-3 the access point generates a random number as a challenge and then waits for the response, the encrypted random number. Now the access point decrypts the response and only if this number equals the formerly generated random number the access point grants network access to the mobile station.

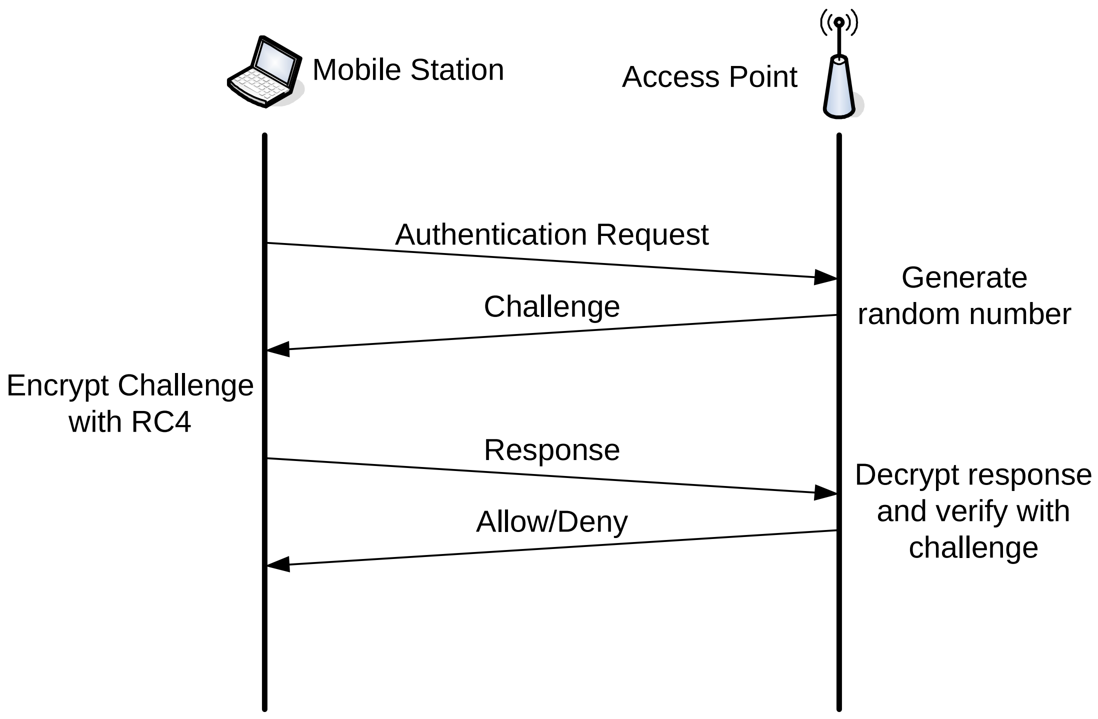


Fig. 8.6-3: Shared Key Authentication

Both cryptographic operations, the encryption at the mobile station and the decryption at the access point, use the RC4 algorithm and need the actual WEP key as an input for the algorithm.

Knowing these details we see that *Open System Authentication* in fact means no security at all and *Shared Key Authentication* only means that both communication partners know the same shared key. A real authentication process is executed neither for the mobile station nor for the access point.

Example 8.6-1: Shared Key Authentication
A mobile station wants to authenticate itself with the *Shared Key Authentication* protocol to an access point. Fig. 8.6-4 shows the message flow for this authentication.

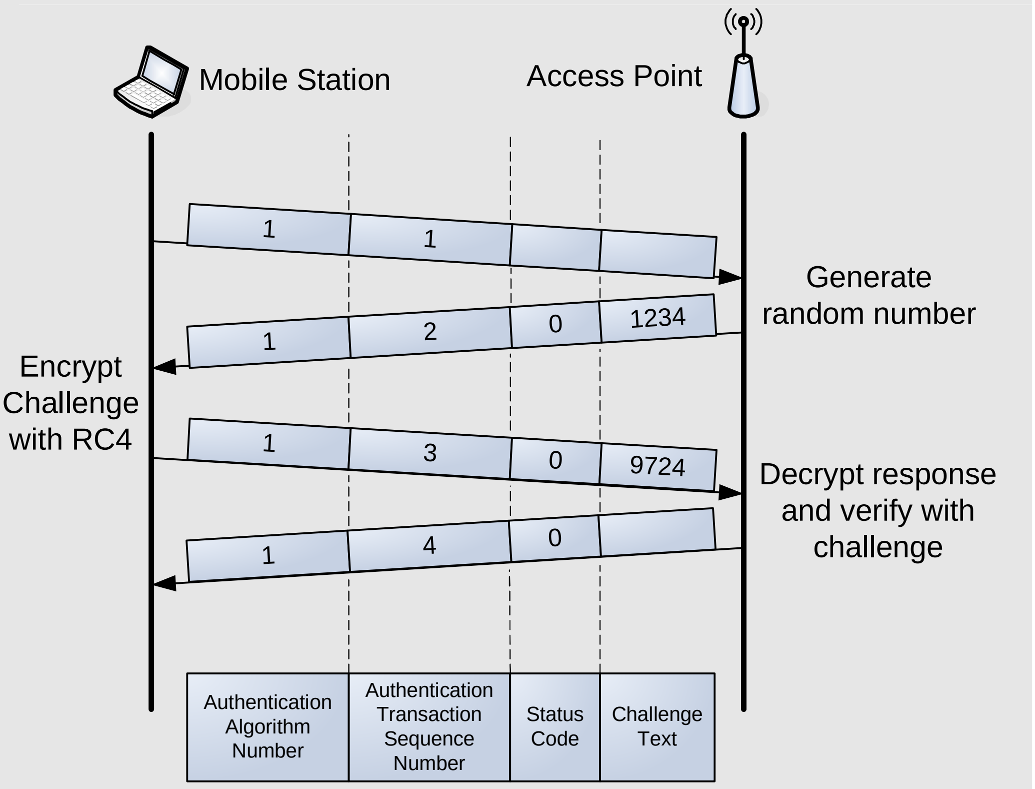

Mobile Station
Access Point
1
1
Generate random number
1
2
0
1234
Encrypt Challenge with RC4
1
3
0
9724
Decrypt response and verify with challenge
1
4
0
Authentication Algorithm Number
Authentication Transaction Sequence Number
Status Code
Challenge Text

Fig. 8.6-4: Shared Key Authentication

The access point generates the challenge `1234` and receives the response `9724`. After checking the response with RC4, the access point sends a *authentication success* message to the mobile station.

Default Keys
Key Mapping Keys

In the specification there are two types of keys defined: **Default Keys**[94] and **Key Mapping Keys**[95]. Both types of keys are used for encryption and have the same characteristics, they

- have a **fixed length** of 40 or 104 bits,
- are **static** and only will be changed after reconfiguration,
- are **shared**, a mobile station and an access point own the same key,
- are **symmetric**, the same key is used for encryption and decryption.

94 Manufactures also use the terms *shared key*, *group key*, *multicast key*, *broadcast key* or just *key*.

95 also: *individual key*, *per-station key* or *unique key*.

Default Keys are used system wide, all access points and mobile stations have the same key. It is possible to define up to four different default keys at the same time – each one with a different key index which is submitted in each frame in clear text (i.e. unencrypted). This is a help for network administrators to change the default keys.

Key Mapping Keys are unique keys for each mobile station. With the knowledge of the MAC address of a mobile station the access point looks up an internal database to choose the unique key for the mobile station. This feature is optional for an IEEE 802.11 compliant access point and only some vendors have implemented this in a few devices.

Unfortunately, the IEEE 802.11 specification does not address out the problem of key management. How are keys made available to all users and access points? Beside non-standard attempts of vendors, like Cisco with LEAP[96], there are no key management techniques provided. The standard answers the question only with this statement:

> *The required secret is presumed to have been delivered to participating stations via a secure channel that is independent of IEEE 802.11.*

8.6.1.2 Privacy and Integrity Mechanisms of WEP

Cyclic Redundancy Check
RC4

There are two security mechanisms defined in the protocol: A ***C*yclic *R*edundancy *C*heck** (CRC) for data integrity and **RC4** (Ron's Cipher[97] 4) for encryption.

Initialisation Vector
XOR

The central cryptographic technique in the IEEE 802.11 standard to support privacy on the wireless interface is the stream cipher RC4. A shared (symmetric) secret key is used to encrypt data at the link layer. The symmetric key ($K_{default}$) is concatenated with a 24 bit ***I**nitialisation **V**ector* (IV) to generate a pseudo-random number stream with the stream cipher algorithm RC4. This key stream is added modulo 2 (*e**x**clusive **OR***) with the plaintext packet, which consists of the payload and a CRC-checksum. Fig. 8.6-5 shows a block diagram of the encryption progress for the sender.

96 ***L**ightweight **E**xtensible **A**uthentication **P**rotocol*, (see Section 8.6.2.2)

97 RC4 was developed in 1987 by Ronald L. Rivest for RSA Data Security Inc.

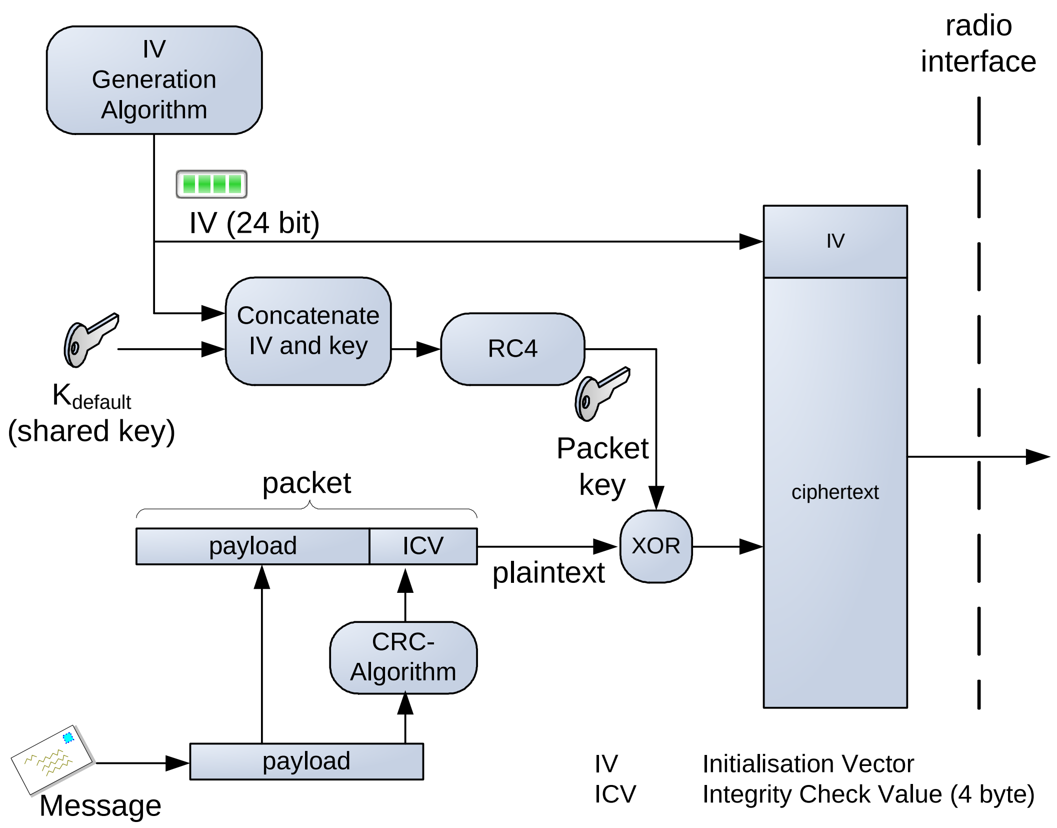


Fig. 8.6-5: WEP encryption
IV - Initialisation Vector
ICV - Integrity Check Value (4 byte)

RC4 encryption algorithm

The **RC4 encryption algorithm** consists of two phases, the key setup and the pseudo-random generation.

Key Setup

The first step in the key setup is the creation of a 256-byte permutation array (S-box). In the beginning the arrayis filled with the values 0-255 in order. Then a second 256-byte array (K-box) is created and filled with the default key. If neccessary the key can be repeated to fill the whole array. Then every element in the S-box is swapped with another element. The following pseudo-code shows this swapping algorithm in detail:

Listing: Initialisation of the S-Box

```
i=j=0
for i=0 to i=255 do
   j = (j + S[i] + K[i] mod 256
   swap S[i] with S[j]
end
```

Pseudo-random Generation

Now the S-box is initialized and the next phase, the pseudo-random generation can be started. This phase consists of more swapping operations and produces one pseudo-random byte per iteration i.

Listing: Key Stream Generation

```
i = ( i + 1 ) mod 256
j = ( j + S[i] ) mod 256
swap S[i] and S[j]
k = (S[i] + S[j]) mod 256
R = S[k]
```

The 802.11 standard was ratified in 1999. At this time export restrictions from the US government for cryptographic technology limited the key size of WEP to 40 bits. The protocol was called 64-bit WEP (40 bit key + 24 bit IV). After abolition of the export restrictions all manufactures implemented an extended version, the 128-bit WEP. The key size was – like in the orginial version – much smaller, because the manufactures added the 24 bit IV (which is always sent in cleartext) to the shared key. The effective key size for 128-bit WEP thus is 104 bit.

8.6.1.3 Weaknesses of WEP

There are several possible attacks against a WEP encrypted wireless LAN. In general these attacks can be divided into active and passive attacks. Passive attacks do not modify the content of network traffic and can be either eavesdropping or traffic flow analysis.

Weak Initialisation Vectors This attack is based on some weaknesses in the key scheduling algorithm of RC4 found by Fluhrer, Mantin and Shamir in 2001 [FMS01] and is also called FMS attack. The attacker has always the IV in cleartext and can analyse the IVs for some special conditions. The found IVs are called "weak", because these IVs result into a relative high correlation between the output and the key. Normally about five to ten million collected encrypted packets are needed to find enough "weak" IVs to reconstruct the key. Newer implementations of WEP avoid these IVs, thus it is harder to collect enough packets.

Dictionary Attack Many access points – and nearly every available operating system – provide a functionality to the user to generate a WEP key (40 bit or 104 bit) from a given password. The used algorithm is normally an MD5-Hash. For a dictionary based attack normally only one packet is needed. Every word (or permutation of word fragments, if additional password hacking tools are used) is translated to a WEP key. With this key and the known IV the packet can be tried to be decrypted. It is not neccessary to decrypt the whole packet, because every WLAN packet has a SNAP-Header (***S****ub* ***N****etwork* ***A****ccess* ***P****rotocol*) which is the same for every IP or ARP packet. If the first bytes of the decrypted packet do not match this header, the rest of the packet also does not.

Korek Attack This attack is named after a programmer with the pseudonym "KoreK" who published a code fragment in 2004. This attack does not need weak IVs anymore, every packet with a unique IV is used in a statistical cryptoanalysis to determine the key. There is a good chance to compute the key with hundreds of thousands rather than millions of packets like in the FMS attack.

Even if the Korek attack does not need so much data input than the attack from FMS, in low traffic networks this could take too long. Active attacks include masquerading, replay, message modification and ***D**enial **o**f **S**ervice* (DoS). There are several known WEP related active attacks like replay attacks or packet injection.

Denial of Service

Replay Attacks If there is not enough traffic in the destination wireless LAN to collect enough packets in an adequate time, the invader can use an active attack against the wireless network to force response messages. WEP does not care about replay attacks where every time the same packet (with the same IV) is received. With no knowlegede of the payload the attacker has to find a packet which forces a response. The easiest way to find such a packet is to guess an ARP[98]-Request. ARP-Request are relativly small and can be identified by the broadcast adress `FF:FF:FF:FF:FF:FF`.

Single Packet Decryption This attack is based on a proof of concept of "KoreK", which he called "chopchop". Because of the XOR operator in the WEP algorithm, each byte in the ciphertext exactly depends on one byte in the cleartext (see Fig. 8.6-5). During the attack the last byte of the ciphered payload $b_{cipher}(n)$ is cut off. Then it is assumed, that $b_{clear}(n)$ is $0, 1, 2, ..., 255$ and for each value the ICV is recomputed and the packet is sent to the access point. If the AP does not drop the packet with the modified byte $b_{clear}(n)$, the packet has the right value and the algorithm continues with $b_{clear}(n-1)$. After thousands of packets (but only a few seconds) the whole cleartext *and* a valid keystream of the RC4 algorithm with a known IV is available.

Packet Injection With a known RC4 keystream and the corresponding IV an attacker is able to inject every possible packet without knowing the actual WEP key.

8.6.2 Wi-Fi Protected Access (WPA)

Because of several security flaws found in WEP, a new and better security standard for 802.11 based wireless networks became neccessary. The development was pushed by the industry group *Wi-Fi Alliance*. This group was founded in 1999 as *Wireless Ethernet Compatibility Alliance* (WECA) and then renamed itself in 2003 to *Wi-Fi Alliance*. The original objective of this alliance was to guarantee the interoperability of different wireless local area network equipment. The missing interoperability was the main lack of the first generation devices, even though the specifications were defined in IEEE 802.11.

Wi-Fi Alliance

98 ***A**ddress **R**esolution **P**rotocol*

Wi-Fi Protected Access

After a delay in the ratification of the standard 802.11i the *Wi-Fi Alliance* announced ***W****i-Fi* ***P****rotected* ***A****ccess* (WPA) as a subset of IEEE 802.11i. This pseudo standard was a change-over from WEP to WPA2 (Section 8.6.3) which implements the full standard IEEE 802.11i. WPA is based on WEP security mechanism but improves it with dynamic keys (Section 8.6.2.1) and better authentification schemes (Section 8.6.2.2).

8.6.2.1 Temporal Key Integrity Protocol (TKIP)

One of the objectives of the *Wi-Fi Alliance* and IEEE 802.11i was to establish improvements in a short time with the constraint, they have to run on legacy (the old WEP) devices. The Initialization Vector is identified as the center of several security problems in WEP, four fundamental weaknesses are listed in Table 8.6-1.

Tab. 8.6-1: *Weaknesses in WEP and their improvements*

Weakness in WEP	Improvement in TKIP
IV is too short, so the same values are reused regularly in a busy network.	The size is increased from 24 bit to 48 bit.
An IV is not related to a station, the same IV can be used with the same key on different stations.	To avoid replay attacks, the IV has a secondary role as a sequence counter.
Some IVs are "weak" (see FMS-attack).	Weak IVs are avoided.
Message Integrity only with CRC-32.	Message integrity is computed by a message integrity protocol.

Temporal Key Integrity Protocol

While designing the ***T****emporal* ***K****ey* ***I****ntegrity* ***P****rotocol* (TKIP) it was obvios that the size of the Initialization Vector must be increased. The working group IEEE 802.11i decided to add 32 bits to the original 24 bits. The total length of 56 bits is reduced in practice to 48 bits to avoid weak keys. Remember, the legacy hardware required a 24 bit IV followed by a 104 bit key to initialize the RC4 encryption engine. The resulting "new" initialization string had a length of 150 bits, but even with firmware upgrades most of the hardware was unable to handle this. This problem was addressed in TKIP as shown in Fig. 8.6-6: the IV is split into two parts. The lower part is padded with 1 byte to avoid known weak keys and the resulting 24 bits are used in the same way as in WEP. The higher part of the IV is merged with the MAC address, the session key and the lower-part IV and results in two key mixing phases into a 104 bits per-packet key.

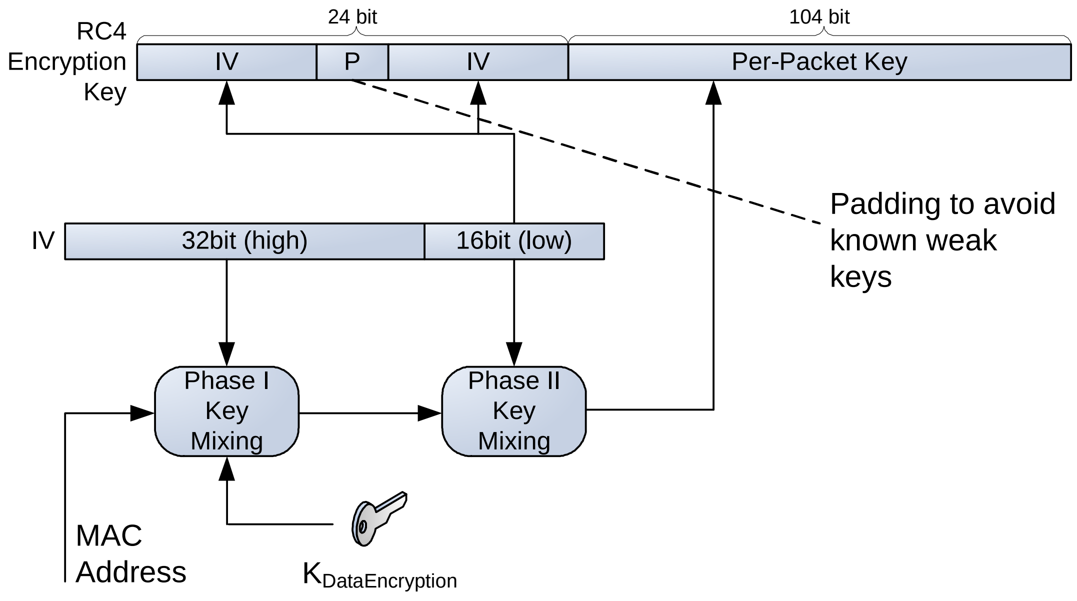


Fig. 8.6-6: Generation of the RC4 encryption key

To protect against replay attacks a mechanism called ***T**KIP **S**equence **C**ounter* (TSC) is used. The TSC is incremented for each packet and it is not possible, that a TSC will be repeated for a given key. A simple way to prevent replay attacks is to ignore all packets with a TSC smaller than the last received message, but actually TKIP uses an advanced solution called **Replay Window**. Normally every packet received is acknowledged separatly. Future standards of IEEE 802.11 aim to allow **Burst-Acknowledgement** where up to 16 packets can be acknowledged with one packet. With this constraint, a packet i only will be accepted if the condition

TKIP Sequence Counter

Replay Window

Burst-Acknowledgement

$$(TSC_{max} - 16) > TSC_i$$

is true, where TSC_{max} is the largest received TSC.

The message integrity in WEP is only checked by a CRC-32 checksum. This checksum is fast to compute, but the linear algorithm is far from a cryptographic secure function. Keeping in mind the limited capacities of the microprocessors inside most of the Wi-Fi devices and the contraint, that improvements have to work with firmware upgrades, it is not possible to use a cryptographic secure hash function like SHA-1[99] or RIPEMD-160[100] for the ***M**essage **I**ntegrity **C**ode* (MIC[101]). Niels Ferguson proposed a method he called *Michael* which uses only fast shift or add operations and no slow multiplications. This simplicity makes *Michael* vulnerable to brute force attacks, but there are countermeasures defined in the protocol. If an invalid MIC is detected the keys for the link get immediately invalid and new keys

Message Integrity Code

99 This algorithm is part of some standardized ***S**ecure **H**ash **A**lgorithms* (SHA).

100 ***R**ACE **I**ntegrity **P**rimitives **E**valuation **M**essage **D**igest* was developed by Hans Dobbertin, Antoon Bosselaers and Bart Preneel in 1996 and produces a 160 bit output.

101 Note that cryptographers use the term MAC (***M**essage **A**uthentication **C**ode*), but this term is used in this context for ***M**edia **A**ccess **C**ontrol*. So MIC is used to avoid confusion.

must be negotiated. This negotiation is delayed 60 seconds, so only every minute an attack can be launched.

To understand the role of *Michael* in the WPA encryption scheme it is necessary to explain the protocol stack in wireless networks in a few words. The ready to send messages generated by operation system functions are called ***M**essage **P**rotocol **D**ata **U**nits* (MPDU) and are forwarded to the MAC layer. If needed, the message is fragmented in this layer and is forwarded to the radio interface. A frame arriving at this interface is called ***MAC** **S**ervice **D**ata **U**nit* (MSDU).

Message Protocol Data Unit

MAC Service Data Unit

The MIC is computed only once for the MSDU and not for every MPDU. This is in contrast to the RC4 encryption where every MPDU is encrypted with a different per-packet key. This results in two advantages. At first the overhead is reduced because the MIC must not to be computed for every fragment. Secondly the computation is done prior to the forwarding to the wireless network adapter, so the computation can be done in the device driver.

The WPA encryption architecture is shown in Fig. 8.6-7 and can be divided into four blocks:

Michael computes a MIC on two 6 byte addresses (source and destination) and the user data (payload). A 64 bit key $K_{DataIntegrity}$ is used for the computation.

Pairwise Master Key

Key Derivation Two different keys are needed: $K_{DataIntegrity}$ for the algorithm *Michael* and $K_{DataEncryption}$ for the RC4 key generation. The ***P**airwise **M**aster **K**ey* (PMK) $K_{PairwiseMaster}$ with a length of 256 bit is generated during the authentication process (see Section 8.6.2.2). Two random numbers and MAC addresses from both the access point and the mobile station are needed to derive the data integrity key and data encryption key.

RC4 Key Generation The RC4 key generation is done in the key mixing phases I and II (see Fig. 8.6-6).

WEP Encryption Each fragment is encrypted in the same way as described in Section 8.6.1.2.

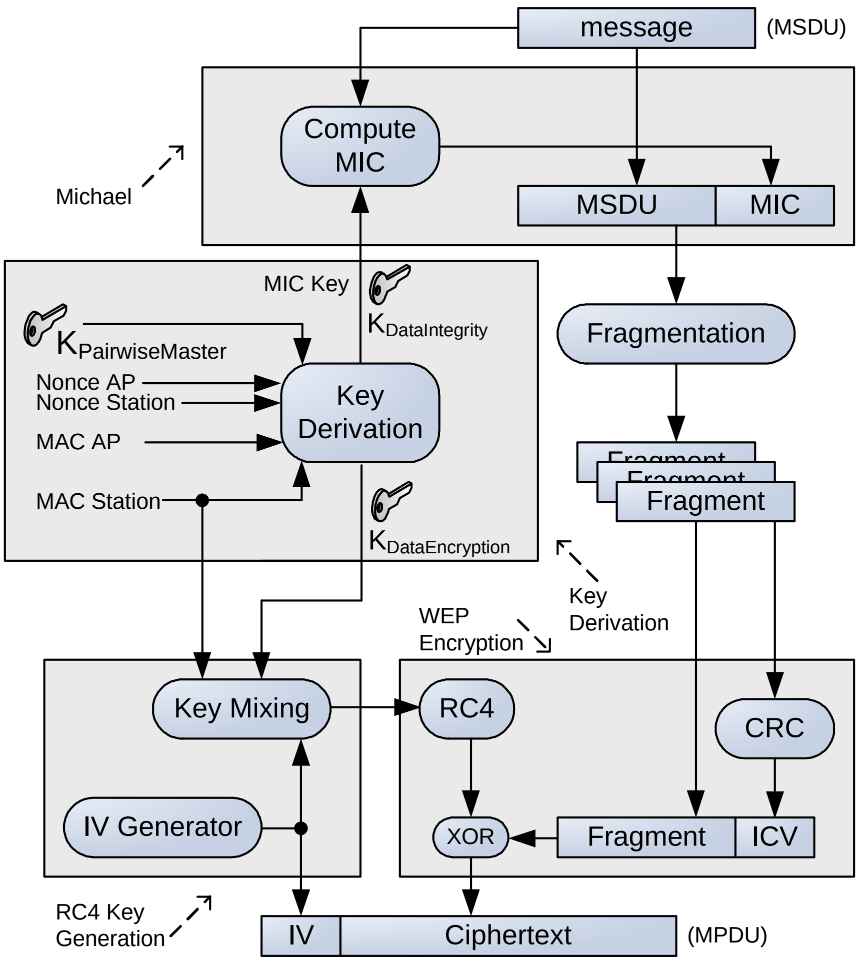


Fig. 8.6-7: WPA encryption

After 2^{16} packets the keys $K_{DataIntegrity}$ and $K_{DataEncryption}$ are changed, this is called rekeying. The key derivation block also generates the ***P****airwise* ***T****ransient* ***K****eys* (PTK) for the authentication process described in Section 8.6.2.2:

Rekeying

Pairwise Transient Keys

- Key for data encryption $K_{DataEncryption}$
- Key for data integrity $K_{DataIntegrity}$
- Key for EAPOW (***E****xtensible* ***A****uthentication* ***P****rotocol*)encryption.
- Key for EAPOW integrity.

Extensible Authentication Protocol

8.6.2.2 Authentication in WPA

WPA (***W****i-Fi* ***P****rotected* ***A****ccess*) provides different authentication schemes. A simple authentication with shared keys or a solution with key management is defined in IEEE 802.1x.

Wi-Fi Protected Access

P*re-Shared* ***K****eys* (PSK) are mostly used in wireless networks in the SOHO (small office/home office) domain. In this domain additional infrastructure like authentication servers cannot be assumed and a shared secret is a sufficient solution for just a few users.

Pre-Shared Keys

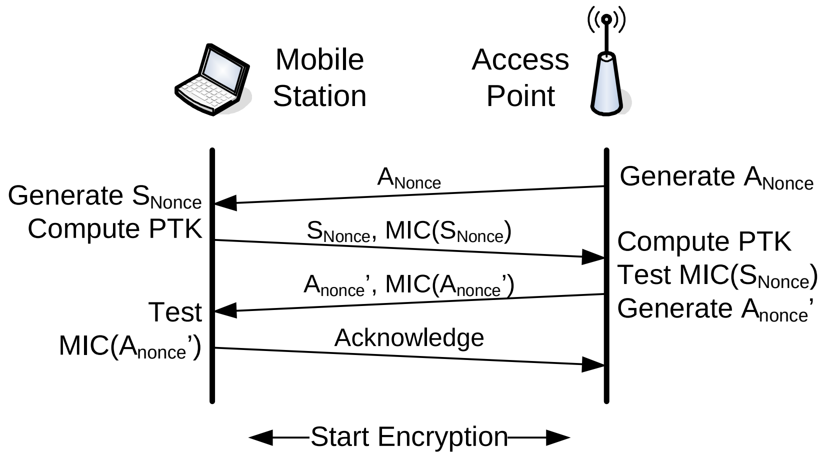


Fig. 8.6-8: 4-Way Handshake with PSK

4-Way Handshake

Fig. 8.6-8 shows the 4-Way Handshake to generate the ***P****airwise* ***T****ransient* ***K****eys* (PTKs) from a PSK[102].

Phase 1 The access point generates a nonce and sends it to the mobile station. There the PTKs will be computed. The nonce is not encrypted or protected, if the message is tampered, the authentication simply fails.

Phase 2 The mobile station generates an own nonce, computes the MIC of the nonce with the PTKs and sends both back to the access point. There at first the PTKs are computed and then the MIC of S_{Nonce} is checked. Only if this check succeeds the authentication process resumes with the following step.

Phase 3 A second nonce with a MIC is sent to the mobile station. Now the mobile station can check if the access point has the ownership of the right PMK.

Phase 4 If the MIC check does not fail an acknowledge will be sent to the access point. After this, all following messages will be encrypted.

Example: Authentication of a mobile station

Fig. 8.6-9 shows the first two steps of the authentication process in a block diagram. The authentication is based on the secret knowledge of the PMK which must be available on both sides. After the mobile station gets the nonce from the access point (and of course the MAC address) it generates the PTK and computes the MIC on a generated nonce. The MIC and the nonce are transmitted to the acces point. In this phase the access point has the same input arguments as the mobile station to compute the PTK: the PMK, A_{Nonce}, S_{Nonce}, MAC_{MS} and MAC_{AP}. With the PMKs the access point can itself generate the MIC in the nonce S_{Nonce}. If both MIC values are the same, the authentication is successful.

102 In this szenario the Pre-Shared Key (PSK) is identical to the Pairwise Master Key (PMK).

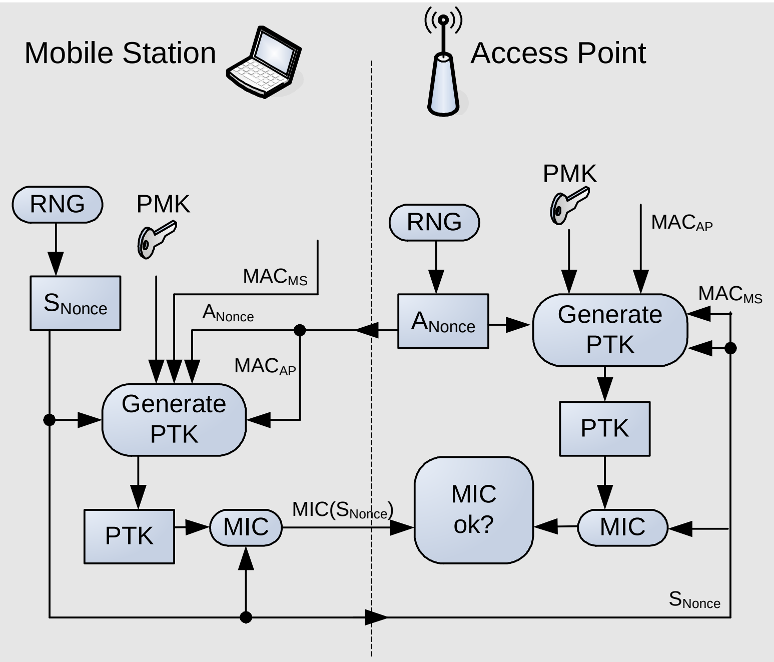


Fig. 8.6-9: 4-Way Handshake with PSK

The standard IEEE 802.1x is a protocol which defines network access control. It is based on the ***E****xtensible* ***A****uthentication* ***P****rotocol* (EAP) and is able to authenticate devices attached to a network. If the authentication of a device failed, no access to the network is possible. The described scheme is very similar to a dial-up situation where a user wants to access a remote network. In these cases the authentication is often done by a ***R****emote* ***A****uthentication* ***D****ial-****I****n* ***U****ser* ***S****ervice* (RADIUS) server, which is optional in WPA.

Extensible Authentication Protocol

Remote Authentication Dial-In User Service

In an EAP based authentication scheme there are three entities acting: The *supplicant* who wants to gain access, the *authenticator* who controls the access and the *authorizer* who decides if access will be admitted. Fig. 8.6-10 shows the EAP for wireless networks, now called ***E****xtensible* ***A****uthentication* ***P****rotocol* ***o****ver* ***W****LAN* (EAPOW).

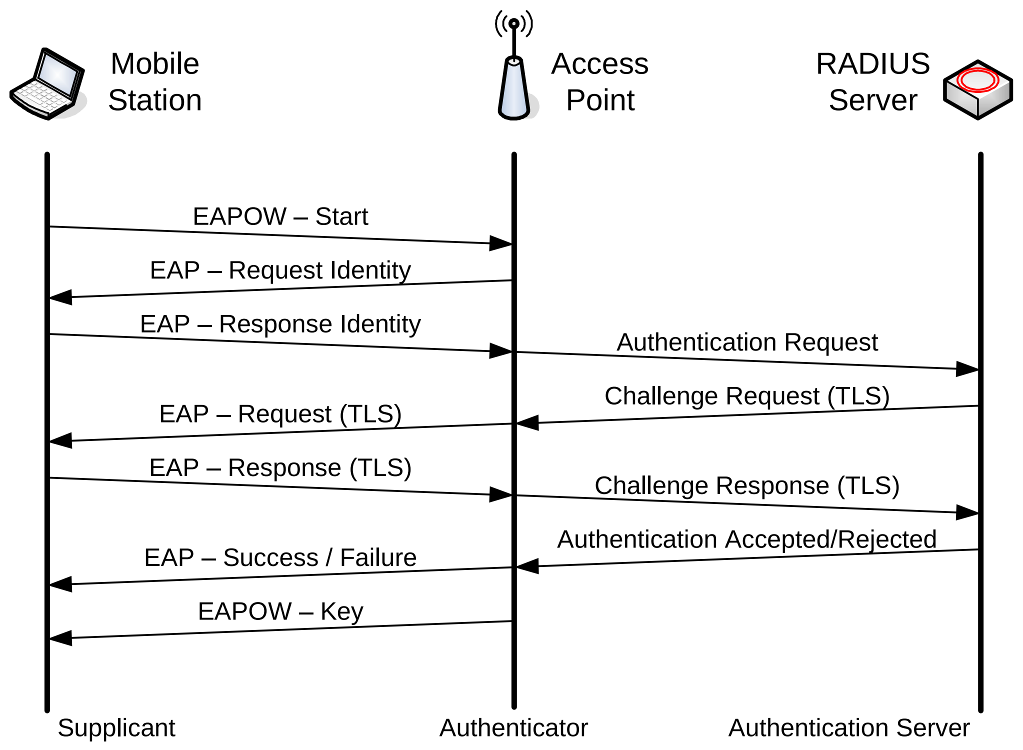


Fig. 8.6-10: EAP over WLAN

There are several authentication schemes available:

- **Message-Digest 5** (MD5): A supplicant is authenticated with the hash value of his password.
- **Transport Layer Security** (TLS): Supplicant and authentication server perform a mutual authentication in this certificate based solution. The same protocol as in the TLS connection phase is used.
- **Tunneled Transport Layer Security** (TTLS) establishes a TLS connection, but the supplicant does not provide a certificate. Other protocols can be used within the TLS tunnel.

Internet Engineering Task Force

- **Protected EAP** (PEAP) is standardized by the ***I**nternet **E**ngineering **T**ask **F**orce* (IETF) and is similar to TTLS.
- **Lightweight EAP** (LEAP) is a proprietary solution from Cisco Systems.

8.6.3 Robust Security Network (RSN)

Robust Security Network

In IEEE 802.11i a new type of wireless network is defined: A ***R**obust **S**ecurity **N**etwork* (RSN). WPA and RSN implement the standard IEEE 802.11i in different depths. Where as WPA only implements a subset, RSN implements the whole standard and is more flexible. Not only TKIP can be used, even an encryption scheme

Advanced Encryption Standard

based on the ***A**dvanced **E**ncryption **S**tandard* (AES) is available. The Wi-Fi Alliance called this WPA2 and makes it mandatory in march 2006 for all Wi-Fi certified products.

Offset Codebook Mode

AES can be used in different modes of operation. With the objective to use an authenticated encryption scheme, the ***O**ffset **C**odebook **M**ode* (OCM) is possible and was discussed in the early stages of the IEEE 802.11i working group. Because of (possible) arising licence fees in the future, the choice for this mode was discarded.

Counter Mode CBC MAC Protocol

Hence a new mode was invented by the working group: ***C**ounter Mode with **C**BC **M**essage Authentication Code **P**rotocol* (CCMP). AES in Counter Mode is also noted as AES/CM[103].

AES is based on the Rijndael algorithm where different block and key sizes of 128, 192 or 256 bit are possible. IEEE 802.11i restricts both values to 128 bit.

Packet Number

A CCMP Header (see Fig. 8.6-11) is added between the MAC header and the payload. A 48 bit ***P**acket **N**umber* (PN) is submitted to protect against replay attacks and as a nonce used in the encryption. The `KeyID` will only be used in multicast traffic or transmissions in mixed groups with TKIP and CCMP.

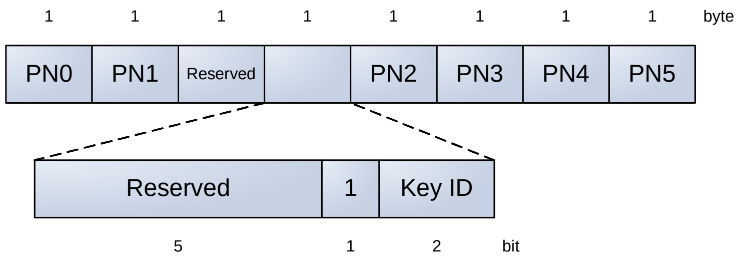


Fig. 8.6-11: ***C**ounter Mode with **C**BC **M**essage Authentication Code **P**rotocol* (CCMP) Header

The first block (Fig. 8.6-12) for the computation of the MIC is constructed in a special way. A nonce is formed by the fields `Priotity`, `Source Address` and `Packet Number`. The `Flag` has a fixed value of `01011001` and the `Length` inidicates the length of the plaintext data.

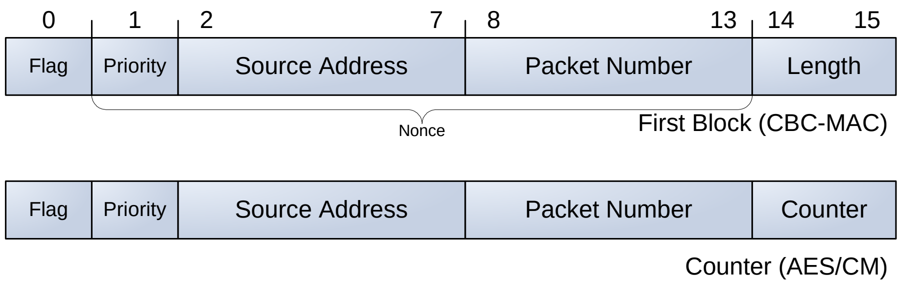


Fig. 8.6-12: First Block (CBC-MAC) and Counter (AES/CM)

The counter for AES/CM is almost identical to the first block of CBC-MAC. To avoid an exactly identical counter the `Flag` is different and a 2 byte internal `Counter` is added. This counter is incremented for each MPDU.

103 Note that we are back in the cryptographers terminology and use in conjunction with CBC the abbreviation MAC for ***M**essage **A**uthentication **C**ode*.

The whole encryption process is shown in Fig. 8.6-13. The MIC is computed over the MAC header[104], the CCMP header and the plaintext data in the block *CBC-MAC* and needs the key, the prior block (or the first block respectively) as input arguments. In the next step the plaintext and the MIC are encrypted with AES/CM.

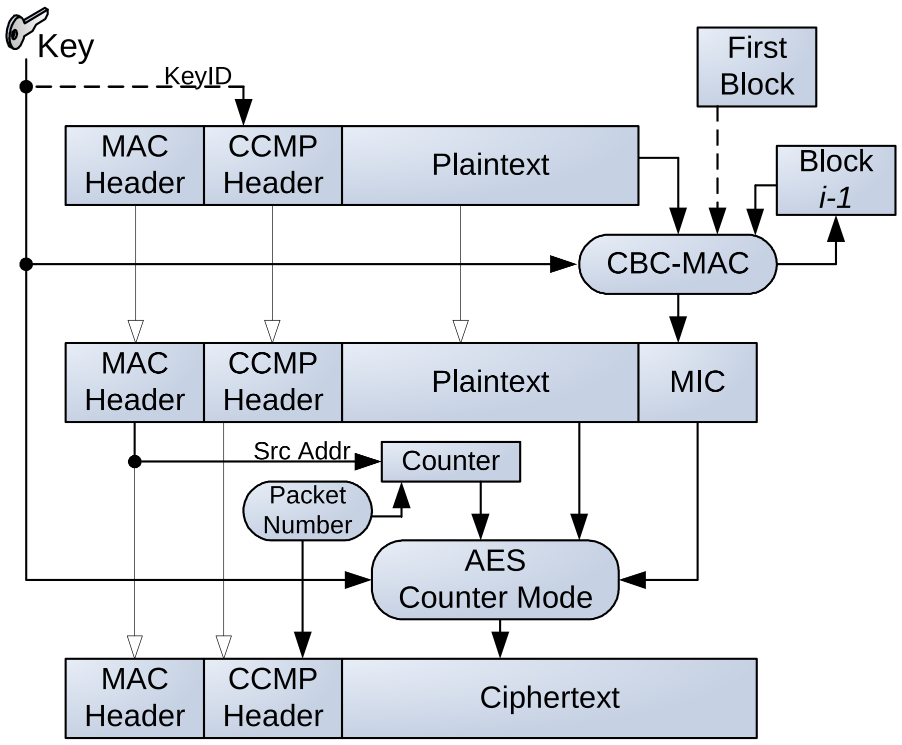


Fig. 8.6-13: CCMP authenticated encryption

8.6.4 Summary

In Table 8.6-2 different security related properties of WEP, WPA (subset of IEEE 802.11i) and WPA2 (whole specification of IEEE 802.11i and also known as RSN) are summarized.

104 Only the immutable fields are included.

Tab. 8.6-2: *Comparison of WEP, WPA and 802.11i*

	WEP	WPA	802.11i (WAP2)
Encryption	RC4	RC4	AES/CM
Key length	56/102 bit	128 bit	128 bit
Data integrity	CRC-32	Michael	CBC-MAC
Header integrity	–	Michael	CBC-MAC
Authentication	Shared Key	802.1x	802.1x
Key management	–	802.1x	802.1x
IV length	24 bit	48 bit	48 bit
Replay attack protection	–	TSC	TSC

None of the original security protocols implemented in IEEE 802.11 are robust. Attackers can easily bypass the authentication process and serious security flaws allow the recovery of the secret encryption key in a short time and the modification and replay of WEP protected packets.

Pre Shared Keys

Service Set Identification

Fortunatly the improvements developed in WPA and RSN protect against the attacks known from WEP. All known attacks against WPA and RSN focus on ***Pre-Shared Keys*** (PSK). The derivation function *KDF2*[105] computes from a passphrase and the **SSID** (***S****ervice* ***S****et* ***Id****entification*) in 4096 iterations a 256 bit hash value, the PSK. There are two possible attacks in a PSK domain:

Pairwise Transient Keys

Internal Attacks Everyone in the same wireless network uses the same PSK. If one captures the data in the WPA 4-Way-Handshake (see Section 8.6.2.2), one has all data to compute the ***P****airwise* ***T****ransient* ***K****eys* (PTK) and reconstruct the transmitted data between the access point and the mobile station.

External Attacks This is a passive attack, where external attackers capture a few (encrypted) packets and try a dictionary attack (see Section 8.6.1.3) against the password hash of the PTK.

The security enhancements between WPA and RSN are fairly small, judged by todays knowledge about security flaws. In both protocols the center of the security level is the authentication type (Section 8.6.2.2). With the right choice, wireless networks with high security requirements can be established. RSN has the benefit of absence of any legacy techniques or compromises. The well known and from many groups checked algorithm AES is used in validated operation modes. In contrast to this the MIC algorithm *Michael* in WPA is relative new and not dicussed in such a detail.

105 *KDF2* is part of the *PKCS #5: Password Based Cryptography* (see [RFC2898])

9 Electronic Payment Systems

9.1 Motivation

With the rapid growth of the Internet, and particularly the World Wide Web (WWW), numerous consumers and organizations participate in a global on-line marketplace. Electronic Commerce is now a large new industry all over the world.

Electronic payment is not a new concept. Actually most of us are familiar with electronic payment: we check our account's balance and tenants transfer rent charge, the fee for pipeline gas, water and electricity is paid to landlord's account via on-line internet services, etc. When setting up an on-line store, one of the most important considerations is the method of payment one intends to use. It is not practical to only accept payment by checks and money orders by mail. This increases the administrative labors to an inacceptable extent, as well as potentially inconveniencing the customers. In today's business arena, flexibility and convenience pay off, and that is why electronic payment systems have become more popular than ever.

Various secure network payment schemes have been developed in universities and different research institutes as well as in commercial organizations. Some of them have undergone small scale testing and some of them have been proven to fail for some reasons (for example, some unconditional privacy protecting systems sometimes could be misused by criminals for blackmailing or money laundering [Solms92]). New technologies, including new security tools, new cryptographic algorithms, new protocols are needed to be developed to protect the privacy during transaction and to make the systems more secure, more efficient and more acceptable by organizations and individuals.

This chapter attempts to introduce the main technologies involved in most payment systems currently available to network users. The cryptological techniques behind most of the schemes supporting their security can be referred from our course materials "Foundations of cryptology".

The characteristics of electronic payment systems are described in Section 9.2. We classify electronic payment systems in Section 9.3 and explain some systems from each category in Section 9.4 and end our chapter by having a bird's eye view on the future of electronic payment systems.

9.2 Characteristics

There are several attempts to describe the characteristics of electronic payment systems [Asokan97], [Medvinsky93]. Most of them from a technology point of view. In this section, we try to describe the main characteristics of general electronic payment systems.

- Applicability. Whether a payment system can be applied to real life and play an important role in paying over open networks is a very important characteristic of an electronic payment system. Whether a payment system is applicable depends on the extent to which customers and merchants accept it. Applicability of a payment system may vary from country to country.

- Usability. Paying with an electronic payment system should not be a complex task.
- Security. Since Internet services are provided today on open networks, the infrastructure supporting electronic commerce, and payment systems in particular, must be resistant to attacks in the Internet environment.
- Reliability. Electronic payment systems should be stable and run smoothly. Otherwise no one would trust and deploy the payment systems.
- Anonymity. Privacy is always an important issue in electronic payment systems. Consumer's identity and personal information should be kept secret. It should not be possible to discover customer's identity or to trace an individual's purchasing activities. For electronic payment systems, anonymity is a necessity to protect consumer's privacy. Some cryptography tools, such as group signature and blind signature, are utilized to keep payment systems anonymous.
- Revokable Anonymity. As we pointed out above, anonymity is a very important issue in electronic payment systems. But under some cases, this characteristic could be misused by economic criminals for blackmailing or money laundering [Solms92]. So, it is strongly recommended to develop anonymity revocable systems. In other words, under some suspicious circumstances, with a trusted third party involved, the anonymity can be removed to discover the identity of the owner. These systems now are called fair electronic payment systems.
- Undeniablility. For electronic payment systems, any transaction which is carried out should not be deniable. This is very natural.
- Scalability. As the commercial use of the Internet grows, the demands placed on payment infrastructure will increases. The payment infrastructure as a whole should be scalable, to be able to handle the additional needs of users and merchants, so that systems will perform normally without performance degradation.
- Interoperability. A payment system is interoperable if it is not dependent on one organization, and is open and allows as many as necessary interested parties to join. This can be achieved by means of open standards for the technology that is used. Examples of interoperable initiatives are the CAFE project [Boly94], and SEMPER project [Semper].

9.3 Classification of Payment Mechanisms: State of the Art

Before we go further into the details of different systems, we present a classification of electronic payment systems. This will be helpful to understand various electronic payment systems.

There are several ways to distinguish electronic payment systems. Based on the way in which money transfer is organized or based on the type of information that is exchanged [Medvinsky93], [Wayner97], existing electronic payment systems can be divided into four groups: Credit Card-based, electronic check, electronic cash and micro-payment systems. Furthermore, electronic cash systems may be distinguished as on-line or off-line systems according to whether banks are involved into transactions during the payment process.

Electronic cash resembles conventional cash, where parties exchange Electronic Tokens that represent values, just as normal banknotes do. While in Credit Card-based systems, bank accounts of customers are transferred over open networks and money is represented by numbers in the accounts. Micropayment system is a special group in which the value of money per transaction is small and fixed with lower security requirements.

We illustrate the classification of electronic payment systems as in Fig. 9.3-1.

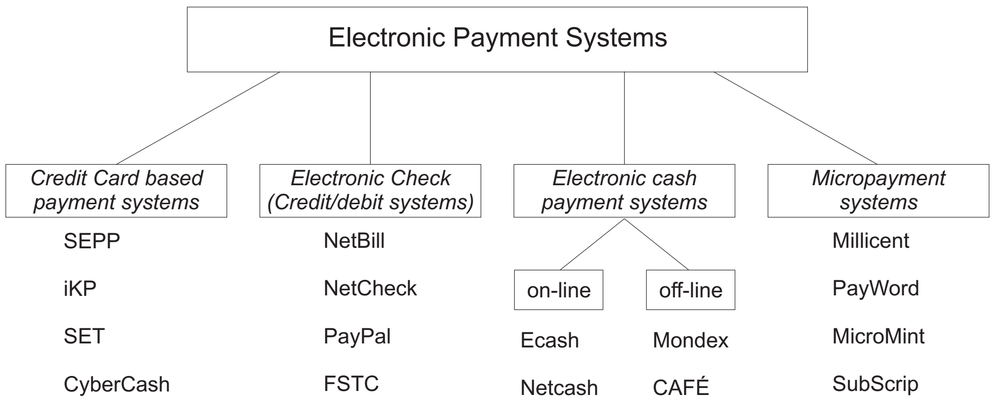


Fig. 9.3-1: Classification of electronic payment systems.

9.4 Overview of Some Typical Systems of Categories

As we mentioned above, electronic payment systems fall into four main groups: Credit Card-based payment systems, electronic check, electronic cash and micro-payment systems.

In this section, we will briefly introduce some important examples from each kind of electronic payment system.

9.4.1 Credit Card-based Payment Systems–SET

To make it more secure, Credit Card-based payment systems always deploy complicated protocols and sophisticated network and cryptographic technologies which make the systems complex and impractical.

Because of combining different cryptographical tools including digital signatures, symmetric and asymmetric encryption techniques, hash functions as well as digital certificates and Public Key Infrastructure(PKI), Credit Card-based payment systems have the advantage of high system security. Credit Card-based payment systems utilize the existing credit card payment method which has been accepted and in use since the early 1960s. On the other hand, because of the use of a mass of computation consuming mathematical calculations, most systems are complicated and inefficient. On-line credit card number checking seems still to be the system's bottleneck. Credit card numbers being openly transmitted over networks is another vulnerability of these kinds of legacy systems.

Various Credit Card-based systems have been introduced. CyberCash [Eastlake96], iKP(the Internet Keyed Payments) [Bellare95], Secure Electronic Transaction(SET) [MC96a], [MC96b], [MC96c] are the most well known systems.

In the CyberCash system, a server acting as a gateway links between the existing financial infrastructure and open networks. CyberCash once planned to improve their system to be SET-compatible.

The iKP has three levels of security with three individual protocols: 1KP, 2KP, 3KP. They differ from each other in the number of involved parties, who hold their own public key pairs. The more the number of parties with own key pairs, the stronger the security the protocol can provide.

In the rest of this section, SET will be described in detail to show as an example how this kind of protocol works. Generally, CyberCash, iKP and SEPP have laid the foundations for the SET protocol.

Background

Working with major credit card providers, a team of IBM researchers in Zurich and Watson Research Center developed a nonproprietary family of secure Internet payment protocols iKP which were mentioned above. They claimed, "We didn't want to develop a competing proprietary protocol, rather, we wanted to create an open, standard protocol that everyone could use".

Based on the framework of iKP, IBM-MasterCard developed a modified protocol called Secure Electronic Payment Protocol(SEPP) by the middle of 1995. Before that, in 1994, Microsoft-VISA jointly designed a protocol, called Secure Transaction Technology(STT), which could safely transfer money across the Internet. Both of these protocols are based on public key cryptography. To be standardized, VISA, MasterCard, supported by IBM and other partners, announced that they had agreed on a convergence of STT and SEPP and developed a new standard for secure paying over the Internet with credit cards in 1996, known as Secure Electronic Transactions(SET).

SET is now an open standard for protecting the privacy and enabling secure credit card transactions over open networks. SET allows all identities to be authenticated via digital certificates and it protects consumer's credit card number from being eavesdropped by combining two complex cryptographical mechanisms - symmetric or private key cryptography and asymmetric or public key cryptography.

Related cryptological techniques

Before turning to the details of SET, some special cryptological techniques are introduced here for better understanding.

- **Dual signature:**Fig. 9.4-1 shows how a dual signature is constructed.

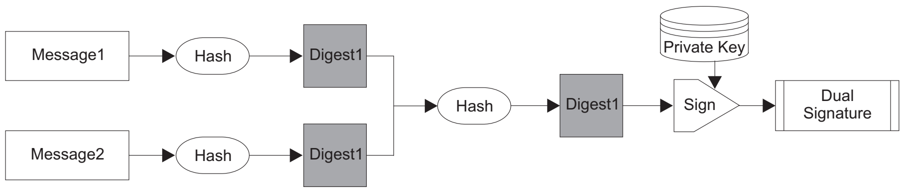


Fig. 9.4-1: Dual signature.

The two messages, message1 and message2, are hashed separately using the same hash function (Md5 or SHA1 etc.). Then the two digests, digest1 and digest2 are concatenated and a new hash value is generated, which is than signed with the sender's private key.

Dual signature can be used to link two related messages without the need to know the message contents. The normal way to generate a dual signature can be described as follows:

– Take two messages $M1$, $M2$. The signer knows both contents.

– Create a normal dual signature: $HH = H(H(M1), H(M2))$. Then get the signature $\{HH\}_{Signer\_private\_key}$ by signing HH with the private key of signer. $\{HH\}_{Signer\_private\_key}$, $H(M2)$ is sent to the holder of $M1$; $\{HH\}_{Signer\_private\_key}$, $H(M1)$ is sent to the holder of $M2$.

– Now, both the holders of $M1$ and $M2$ can verify the signature by regenerating the HH because they know $M1$ and $M2$ respectively. But the holder of $M1$ will not know the message contents of $M2$ and the holder of $M2$ will not learn the message contents of $M1$.

The use of dual signature in SET can be seen later.

- **Digital envelope:** If public key systems are used only for message authentication, then a simple digital signature using the signer's private key with a hash function can do that. However, if public key systems are used as an encrypting tool to achieve confidentiality, then a technique called digital envelope can be applied to achieve that goal.

  Fig. 9.4-2 shows the process of digital envelope.

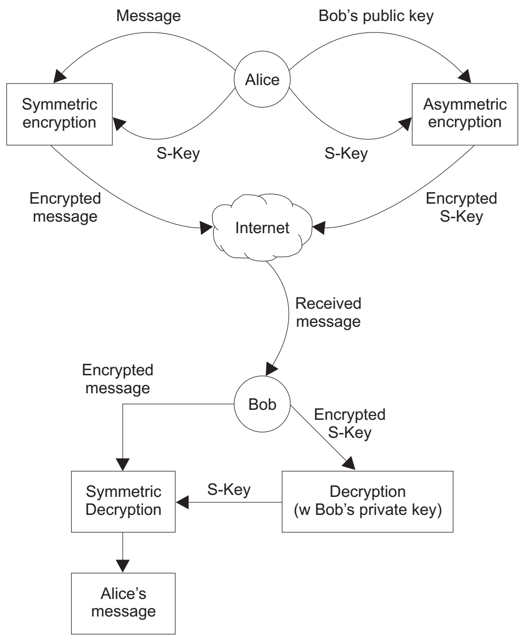


Fig. 9.4-2: Construction of a digital envelope.

The digital envelope consists of a message which will be encrypted using private key cryptography and a secret key which will be encrypted using public key cryptography. The following describes the process of creating a digital envelope:

As in the diagram, to generate a digital envelope, Alice chooses a secret key and encrypts the message with it, then encrypts the secret key using Bob's public key. She sends Bob the encrypted message concatenated with the encrypted secret key. When Bob wants to read the message, he decrypts the secret key, using his private key, and then decrypts the message, using the secret key.

The reason for using a digital envelope is that public key cryptography systems have stronger security but lower encrypting speed (from a software point of view, public key systems are about 1000 times slower than private key systems).

Most of the messages in SET are transmitted using the digital envelope technique.

- *Extra strong encryption* In SET, instead of encrypting using the digital envelope technique in some special cases, data is directly encrypted with RSA public key algorithm to enhance the security. This process is called extra strong encryption.

Framework

The framework of SET is shown in Fig. 9.4-3.

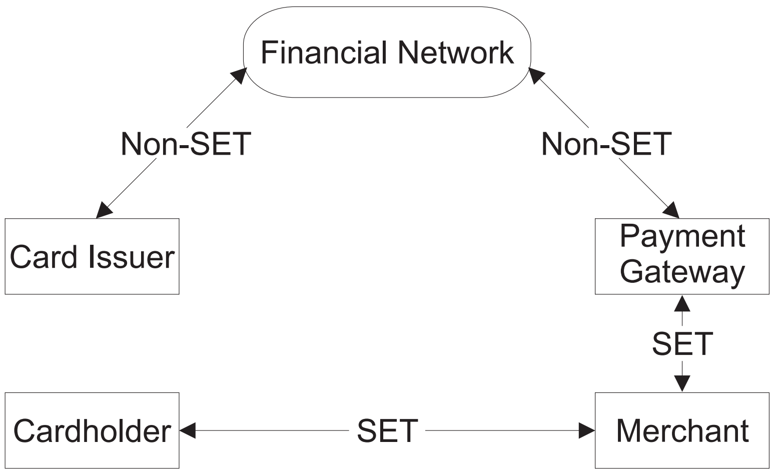


Fig. 9.4-3: Framework of SET.

It can be seen from the diagram that SET is used between cardholder, merchant and payment gateway. The payment gateway acts as a front end to the existing financial network. SET was designed only with regard to the payment process instead of being a general purpose E-commerce protocol. In other words, some processes, for example, price negotiation and delivery protocol, have not been considered in SET.

Authentication model

Each of the parties involved in SET will be required to authenticate themselves at some points in the payment process. The private key of a public-key pair is used as the proof to authenticate the owner's identity and the public key part must be certified by a trusted third party.

Digital certificate(X.509 version 3 format [ITU97]) is the tool used in SET to bind identities with their corresponding public keys. The identity given by the *subject* field in a certificate is bound to the public key held in *subjectPublicKeyInfo* field and this binding is certified by the *issuer* by generating a digital signature on the certificate using the private key of the *issuer*.

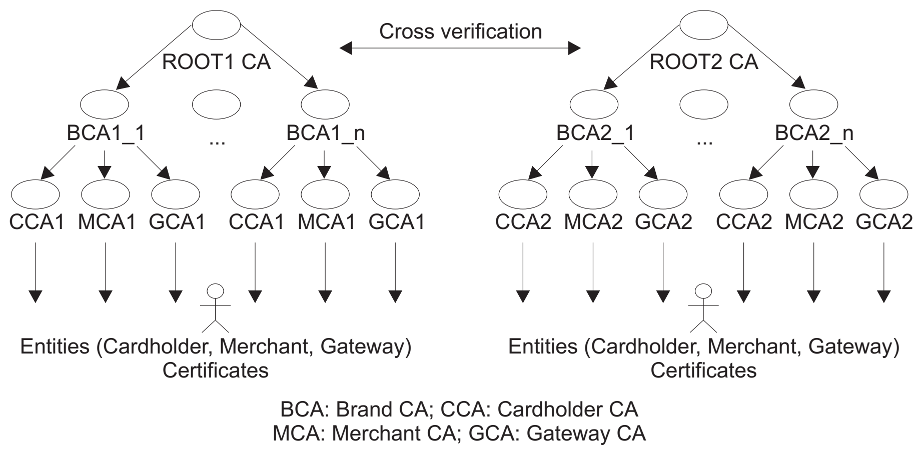

Cross verification
ROOT1 CA
ROOT2 CA
BCA1_1
...
BCA1_n
BCA2_1
...
BCA2_n
CCA1 MCA1 GCA1 CCA1 MCA1 GCA1
CCA2 MCA2 GCA2 CCA2 MCA2 GCA2
Entities (Cardholder, Merchant, Gateway) Certificates
Entities (Cardholder, Merchant, Gateway) Certificates
BCA: Brand CA; CCA: Cardholder CA
MCA: Merchant CA; GCA: Gateway CA

Fig. 9.4-4: Certification authority hierarchy in SET.

A certificate authority(CA) maintained by a trusted third party is responsible for certificate issuing and management. Fig. 9.4-4 shows the certification architecture used in SET.

Two kinds of certificates exist in SET payment processes: one of them is the signature certificate where the corresponding private key is used to sign messages; another is the data encipherment certificate where the public key is used to encrypt messages or to exchange keys. All parties must obtain certificates from a certification authority (CA) before starting using SET to make transfers over networks.

Message structure

A typical SET transaction is shown in Fig. 9.4-5. The messages needed to perform a complete purchase transaction usually include:

- Initialization(PInitReq/PInitRes)
- Purchase order(PReq/PRes)
- Authorization(AuthReq/AuthRes)
- Capture of payment(CapReq/CapRes)
- Cardholder inquiry(InqReq/InqRes)[only optional]

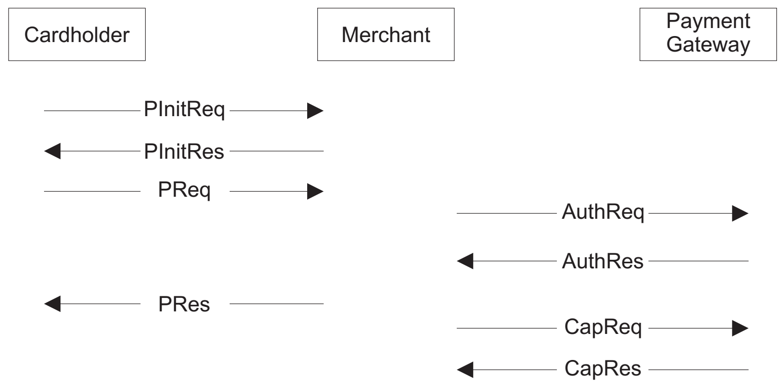


Fig. 9.4-5: Main messages exchanged in a SET transaction.

All the messages are defined in a hardware-independent format in SET specifications to enhance interoperability. Some parts of the messages, such as the financial data about a credit card and purchase order are encrypted so that the card number will not be revealed to the merchant and the order information will not be learned by the payment gateway.

In this section, we explain an example to show how to perform a SET transaction. More details can be found in the SET specifications [MC96a], [MC96b], [MC96c].

As shown in Fig. 9.4-6, there are three parties in a SET system: a wallet software installed on the user's machine; a merchant server running at the merchant's Web site; and a payment gateway usually running at a financial organization (the merchant's bank).

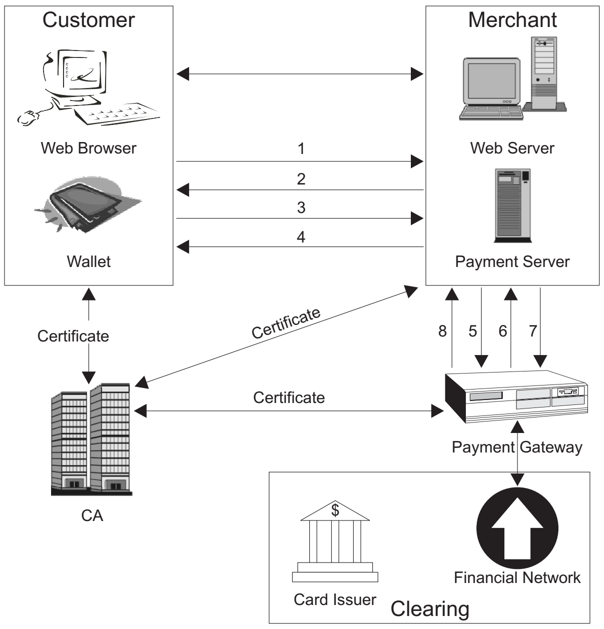


Fig. 9.4-6: A typical SET transaction.

To perform a SET purchase, a user selects products from the merchant's Web site and then the merchant sends back a response to wake up the user's wallet program. The user selects which credit card she wishes to use to pay with and then the wallet automatically performs a SET payment protocol by exchanging messages with the merchant and payment gateway to proceed the payment as follows:

- The user sends a PInitReq message to the merchant as shown in Fig. 9.4-7. In PInitReq message, BrandID is the brand of card that will be used in the payment(VISA or MasterCard); LID_C is a local transaction ID; Chall_C is used to guarantee the freshness of the communication; Thumbs is a list of certificates already stored by the user to verify the user's certificate chain which links back to the root certificate.

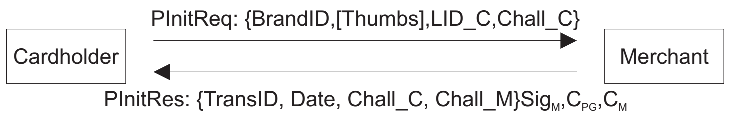

PInitReq: {BrandID,[Thumbs],LID_C,Chall_C}
Cardholder
Merchant
PInitRes: {TransID, Date, Chall_C, Chall_M}Sig_M,C_PG,C_M

Fig. 9.4-7: Initialization messages.

- On receiving the PInitReq message from the user, the merchant will generate PInitRes. In the merchant's response message, TransID is a globally unique ID combined with the LID_C. A new challenge, Chall_M, generated by the merchant is included along with the challenge Chall_C from the user. In addition, the needed certificates and the current date are also included in the PInitRes message.
- Upon receiving of the PInitRes from the merchant, the user will generate purchase request(PReq) message as shown in Fig. 9.4-8. PReq is the most complex message in the payment protocol. It consists of two parts: the order information(OI) and payment instructions(PI).

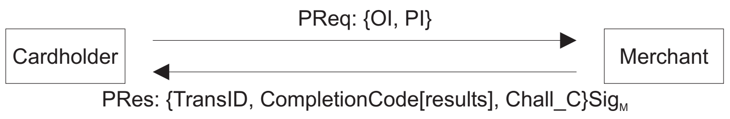


Fig. 9.4-8: Purchase request and response messages.

Each part is encrypted separately. The OI holds data that identifies the order description and can be seen only by the merchant. The PI which contains credit card number, purchase amount, etc., is encrypted with the payment gateway's public key and the merchant has no access to this part. Later, The PI will be forwarded to the payment gateway as part of the authorization. The constructions of OI and PI are shown in Fig. 9.4-9.

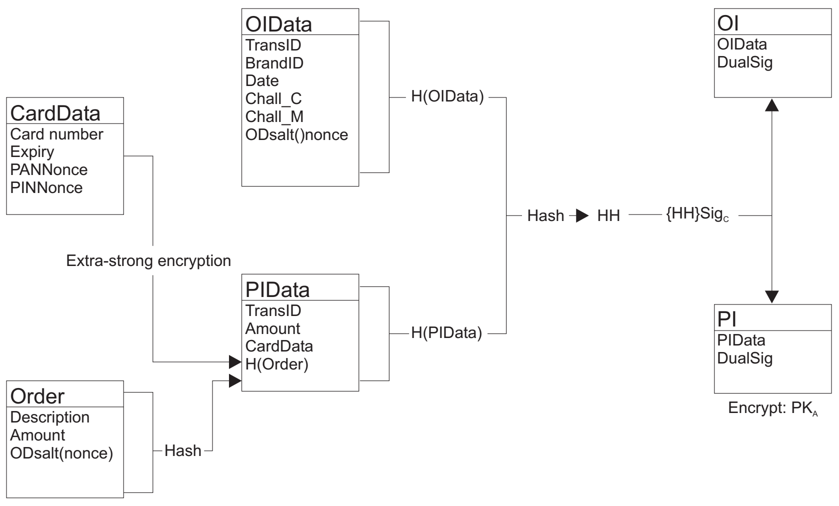


Fig. 9.4-9: Constructions of OI and PI.

OIData consists of data from the initialization phase. ODsalt is a random nonce to prevent dictionary attacks[106]. PIData contains the confidential information on the user's credit card. PIData is encrypted using the extra strong encryption technique so that it can be seen only by the payment gateway. The merchant has no access to these data. A dual signature(DualSig), generated using the hashes of OIData and PIData(the data contents of OI and PI), links order information to the authorization payment instructions by showing that the user and the merchant signed together and agreed on that. Then anyone, who possesses either OIData and PIData can verify the signature without having to know the other. $H(order)$ is meant for the gateway to identify the order which the user is referring to without giving the order details and the gateway can compare this value with that from the merchant to make sure that the order is the original one from the user.

- When the merchant receives the purchase request from the user, the OI and PI parts are extracted. The purchase request message can be verified with the dual signature on OI using the user's certificate. A purchase response message(PRes) can be sent immediately or any time later in the protocol. The contents of the message will be affected by the stage of processing in the protocol at which the purchase response is returned from the merchant (e.g. order received, transaction authorized or transaction captured). The user can check the state of the payment by sending inquiry messages to the merchant. Before sending a PRes to the user, the merchant will normally perform the authorization and capture steps of the payment.
- Now as shown in Fig. 9.4-10 the merchant sends authorization request (AuthReq) to require the gateway to verify that the user has credit for the purchase and obtains permission from the card issuer to charge the transaction. The construction of AuthReq is shown in Fig. 9.4-11.

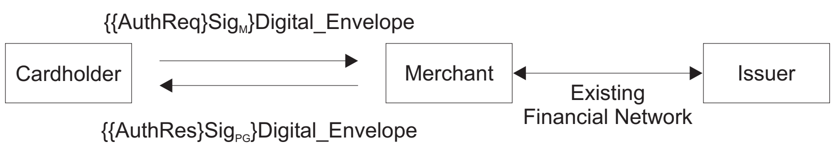

{{AuthReq}Sig_M}Digital_Envelope
Cardholder
Merchant
Issuer
Existing
Financial Network
{{AuthRes}Sig_PG}Digital_Envelope

Fig. 9.4-10: Authorization request and response messages.

106 That is, the nonce stops malicious users from guessing the hash value $H(OIData)$, by trying all possible combinations of dictionary words in the order description

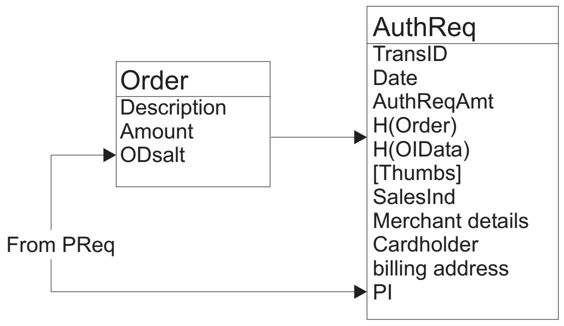


Fig. 9.4-11: Construction of AuthReq.

AuthReq message is signed and encrypted using the digital envelope technique by the merchant. Similar work can be done on the authorization response message (AuthRes).

- After receiving AuthReq from the merchant, the payment gateway decrypts and verifies the message. If the hash $H(order)$ contained in AuthReq matches with the one present in the PI from the PReq message, the gateway confirms that both the merchant and the user agreed on the order. The $H(OIData)$ and the dual signature in the PI prove that the order is originally from the user without revealing any order details. Then the gateway tries to obtain the authorization through the existing financial network. Having received an authorization from the card issuer, an authorization response (AuthRes, as shown in Fig. 9.4-12) containing a capture token, which is used later by the merchant to capture the payment, is returned to the merchant. Upon receiving the authorization response, the merchant makes sure that the card issuer has confirmed the card details and can ship purchased goods to the user.

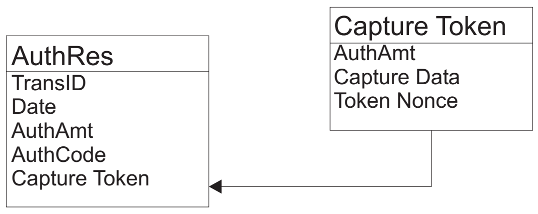


Fig. 9.4-12: Construction of AuthRes.

- Having processed the order from the user, now the merchant sends a capture request (CapReq) message to the payment gateway to move the transaction to the capture state and request the previously authorized payment to be transferred to her account. A single request can contain several authorized payments as shown in Fig. 9.4-13.

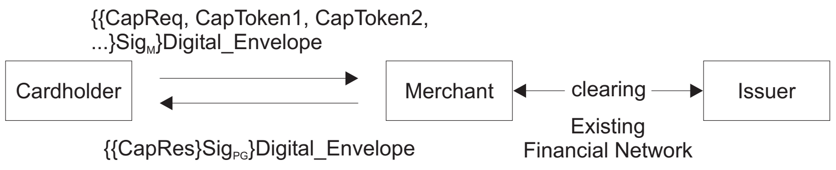

{{CapReq, CapToken1, CapToken2, ...}Sig$_M$}Digital_Envelope
Cardholder
Merchant
clearing
Issuer
Existing Financial Network
{{CapRes}Sig$_{PG}$}Digital_Envelope

Fig. 9.4-13: Capture request and response messages.

The construction of the CapReq is shown in Fig. 9.4-14. In each token, the corresponding authorized amount and transaction ID (TransID) are included. At this point, the merchant will be credited and transaction fees may be deducted at this stage.

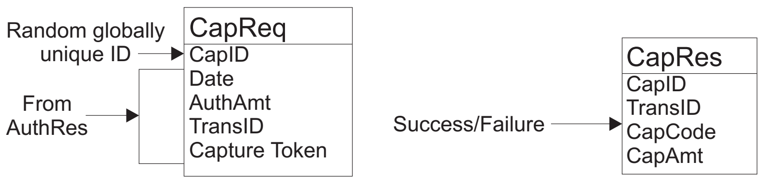


Fig. 9.4-14: Construction of capture request and response messages.

- After verifying the capture request, the capture response (CapRes as shown in Fig. 9.4-14) containing a success indication, the settled amount, and a capture code from the financial network will be returned to the merchant. Both CapReq and CapRes messages are signed and encrypted using the digital envelope technique.

After a success capture, now a typical SET transaction is complete. More details and descriptions can be found in SET specifications [MC96a], [MC96b], [MC96c].

SET or SSL

Backed by MasterCard, Visa and IBM, SET is a completely secure card payment protocol using strong encryption and authentication schemes. Merchants do not have access to the consumer's credit card.

Unfortunately, SET is too complicated to be adopted by people all over the world. SET takes only credit cards as payment tools and needs to install the wallet software on the user's machine. Furthermore, a trusted third party CA must be maintained to authenticate all parties involved in the transactions. A trusted CA issuing certificates to users around the world is very difficult. In contrast to SET, another protocol, called Secure Sockets Layer (SSL) [Kocker97] developed by Netscape is currently the most widely used tool for performing secure transactions over the Internet and is supported by most Web servers and clients including Netscape Navigator and Microsoft Internet Explorer. Although SSL is a secure protocol for more general purposes, when it is deployed in electronic payment systems, SSL is more simple and flexible with higher efficiency. However the merchant does have access to the confidential details of the consumer's credit card in the SSL protocol.

SET or SSL? Which will be the winner is still not quite clear. It seems currently that SSL takes a lead when competing with SET in the electronic payment field.

9.4.2 Electronic Check–NetBill

In the electronic check-like payment model (which can also be called debit-based payment), such as NetBill [Sirbu95] and NetCheque [Neuman95], etc., the systems work much like the check approach in our real life. The biggest difference to Credit Card-based payment systems is that, in Credit Card-based payment systems, all that is needed is a credit card number. The most important aspect in Credit Card-based payment systems is how to protect the consumer’s credit card number. In electronic check systems, electronic check works much like a conventional paper check containing a consumer’s signature. Unlike a paper check however, it has to be endorsed by the merchant before it is paid.

In this section, NetBill will be introduced as an example to show how this kind of electronic payment systems work.

Architecture of NetBill

NetBill [Sirbu95], [Cox95] is an electronic check-like system developed at the Carnegie Mellon University. The developers designed NetBill similar to a check in that immediate transfers from one personal account to another takes place at the time of the purchase.

The architecture of NetBill is shown in Fig. 9.4-15.

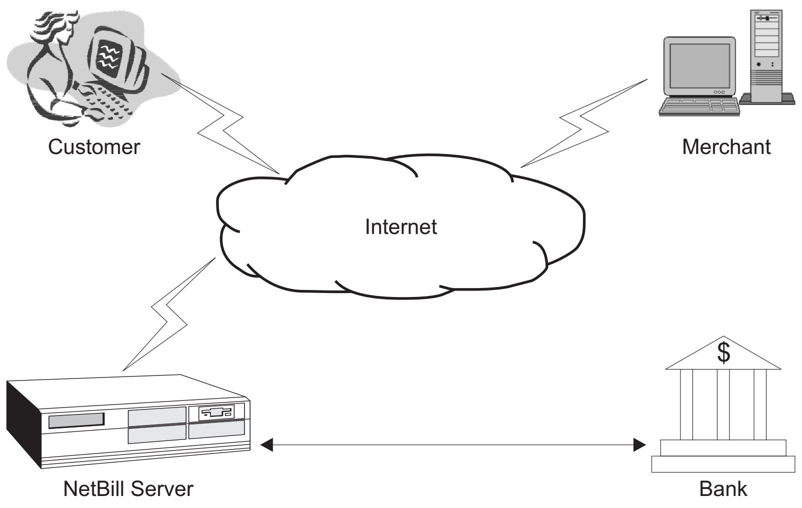


Fig. 9.4-15: Architecture of NetBill.

NetBill system consists of three participants: customer, merchant and a NetBill server which maintains accounts for the merchant and the customer. These accounts are linked to existing financial institutions. After a purchase (of digital goods), the customer’s account is debited and the merchant’s account is credited. Later, the NetBill server will clear the purchase with the conventional financial institution by transferring funds from the customer's account in her bank to the merchant's bank account.

NetBill provides transaction support through libraries integrated with different client-server pairs. The client library is called the *checkbook* and the server library is called the *till*. These libraries use a single transaction protocol to communicate between client and NetBill server. All communication between two parties is encrypted to protect it against attackers.

Public Key Kerberos

With regard to the Netbill payment system the secure distribution of the symmetrical encryption keys in order to encrypt and decrypt messages during the transactions is an important issue. NetBill uses a modified version of pure Kerberos [Steiner88] scheme called Public Key Kerberos [Sirbu95], [Sirbu96]. The modifications retain the efficient symmetric encryption scheme but decrease the Kerberos server dependency by using the public key system during the protocol exchanges.

In the traditional Kerberos system, there is a Key Distribution Center (KDC), which is responsible for issuing tickets for communication parties. All parties have a shared secret key with the KDC so that all transactions between any party and the server are protected by encryption using the shared key. If A wishes to send a message to B, she will first get a ticket (T_{AB}) from the KDC containing the session key (K_{AB}) which will be used to encrypt and decrypt messages exchanged between A and B. K_{AB} is encrypted using the shared key between B and the KDC. Also, A's identity is included in T_{AB}. T_{AB} will be encrypted using the shared key between B and the KDC so that only B can encrypt the ticket and get the session key K_{AB}. Now, A can send messages to B as follows:

$$T_{AB}, \{Message\}_{K_{AB}}$$

When B receives this, she can extract the session key K_{AB} from the ticket using the shared key with the KDC and can decrypt the ciphertext to get the original message from A. In this way, A's identity is also authenticated.

The NetBill system differs to this in that KDC does not exist. Instead, before any communication takes place, A gets ticket T_{AB} directly from B using the public key cryptographic technique. First, A creates a symmetric encryption key, $K_{Challenge}$, and sends the following message to B:

$$[A, B, TimeStamp, K_{Challenge}]_{K_{OneTime}}, [K_{OneTime}]_{PK_B}, Sig_A$$

It is assumed that both A and B have got each other's public keys previously. On receiving the message, B can verify the signature generated by A using the public key of A. B can also use her private key to reveal $K_{OneTime}$ so that she can decrypt the contents which contain $K_{Challenge}$. Now B can construct a normal Kerberos ticket, T_{AB}, and the session keyK_{AB} and send $[T_{AB}, K_{AB}]_{K_{Challenge}}$ back to A by encrypting it with $K_{Challenge}$ which can only be shared with B, the holder of the private key SK_B. A unwraps the message using $K_{Challenge}$ and makes sure that T_{AB} and K_{AB} are generated by B.

All communication parties have to obtain a ticket from the other before messages are exchanged. In NetBill, T_{CM} is the ticket established between the customer and the merchant. Similarly, T_{MN} is the ticket between the merchant and the NetBill server. T_{CN} is the ticket between the customer and the NetBill server.

NetBill transaction protocol

As shown in Fig. 9.4-16, there are at least eight steps in the NetBill transaction protocol as follows:

- The customer accesses the merchant's web server and chooses items. Then customer's client application indicates to the *checkbook* library that it would like a price quote from a particular merchant for a chosen product. The *checkbook* then sends a request as follows for a quotation to the merchant's *till* library.

  $$T_{CM}, [Credentials, PRD, Bid, RequestFlags, TID]K_{CM}$$

  The *credentials* element specifies any group memberships that may merit a discount. The product request data(*PRD*) describes the goods required and the *Bid* is the price being offered by the customer. *TID* is a unique transaction ID.

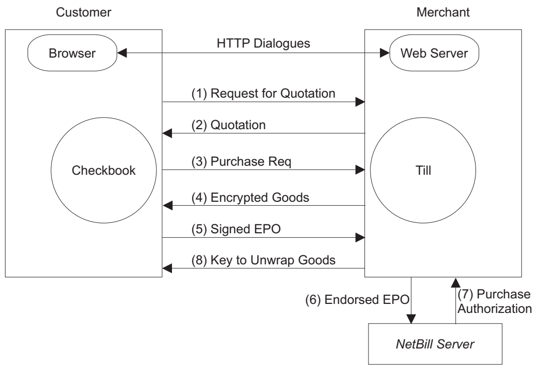


Fig. 9.4-16: Transaction protocol.

- The merchant, on receiving the request, invokes an algorithm to determine a price for the customer and returns a message as follows to the customer:

  $$[ProductID, Price, RequestFlags, TID]K_{CM}$$

  The merchant stores the *PRD* for later use in delivering the goods, generates a new set of *RequestFlags* based on its response to the customer's *RequestFlags*, and generates a *price* quote. Also the transaction Id (TID) is included to link the quotation back to the original request made by the customer. The ProductID returned by the merchant is a human-readable product description that will appear on the customer's NetBill statement.

- If the customer accepts the price quoted, she invokes the *checkbook* to send a digital signed purchase request to the merchant's *till*. Alternatively, the customer can specify that her client application accepts any price below a specified amount and send a purchase request automatically.

  Once the customer and merchant have negotiated a price for the goods in question, the customer directs the merchant to deliver the goods by supplying the TID that was used in the price request phase:

  $$T_{CM}, [TID]K_{CM}$$

- After receiving the purchase request, The merchant generates a unique symmetric cipher key K_{goods}, encrypts the goods using this key and sends the encrypted goods to the customer, along with a cryptographic checksum computed on the encrypted goods. The customer can verify the integrity of the goods by applying the secure hash algorithm to the encrypted goods and check whether it matches with the one computed by the merchant to detect any discrepancy before proceeding further. The merchant also sends an Electronic Payment Order *ID*, or *EPOID*, with the goods.

  $$[Goods]K_{goods}, [SHA([Goods]K_{goods})]K_{CM}, EPOID$$

  The *EPOID* is a globally unique identifier which will be used in the NetBill server's database to uniquely identify the transaction. It consists of three fields: a field identifying the merchant, a timestamp marking the time at the end of the goods delivery, and a serial number to guarantee uniqueness.

- Upon receipt of the encrypted result, the customer verifies the checksum. Until now, she can not decrypt the result and neither has she been charged. Then the customer sends back a signed electronic payment order (*EPO*) back to the merchant's *till*.

  $$T_{CM}, [EPO, Sig_C]K_{CM}$$

  Before the EPO is submitted, the customer may cancel the purchase at any time.

  An *EPO* consists of two sections, a clear part containing transaction information which is readable by the merchant and the NetBill server, and an encrypted part containing payment instructions which is readable only by the NetBill server. The clear part of the *EPO* includes:

  – The customer's identity

  – The Product ID and price specified in the merchant's quotation

  – The merchant's identity

  – The cryptographic checksum of the encrypted goods

  – The globally-unique EPOID

The encrypted part of the EPO includes:

– A ticket proving the customer's true identity

– The customer's NetBill account number

– A customer memo field

- After the customer presents the signed *EPO* to the merchant, the merchant verifies the customer's signature, checks that the product ID, price, and goods checksum are in order and then the merchant endorses it and forwards the endorsed *EPO* to the NetBill server. The endorsed *EPO* adds the merchant's account number $MAcct$, the merchant's memo field $MMemo$, and the goods decryption key K_{goods}, as well as the merchant's signature Sig_M:

$$T_{MN}, [(EPO, Sig_C), MAcct, MMemo, K_{goods}, Sig_M]K_{MN}$$

- The NetBill server makes its decision based on verification of the signatures, the privileges of the users involved, the customer's account balance, and the uniqueness and freshness of the EPOID. It then issues a *Receipt* containing the result code, the identities of the parties, the price and description of the goods, the EPOID, and the key K_{goods} needed to decrypt the goods. The *Receipt* is digitally signed by the NetBill server. The *Receipt* is denoted:

$$Receipt = [ResultCode, C, Price, ProductID,$$

$$M, K_{goods}, EPOID]Sig_N$$

  The NetBill server also will include some account status information with *Receipt* and sends the following to the merchant:

$$[Receipt]K_{MN}, [EPOID, CAcct, Balance, Flags]K_{CN}$$

- The merchant unwraps the *Receipt*, keeps a copy and re-encrypts it using K_{CM} and then forwards the message to the customer:

$$[Receipt]K_{CM}, [EPOID, CAcct, Balance, Flags]K_{CN}$$

- Finally, the customer extracts K_{goods} from the *Receipt* and decrypts the goods that were delivered earlier. The payment now has been made.

NetBill provides a total electronic payment system from price negotiation to goods delivery with strong security. But the NetBill server has to be involved in all transactions. This can lead to a bottleneck of the whole network system.

NetCheque [Neuman95] is a another check-like electronic payment system developed at the Information Sciences Institute of the University of Southern California.

NetCheque is a distributed accounting service supporting the credit-debit model of payment. Unlike NetBill, NetCheque is completely based on the use of Kerberos instead of using the Public Key Kerberos. Because of the efficiency of the symmetric key encryption algorithm, NetCheque is more computationally efficient than other systems based on public key cryptography.

The Financial Service Technology Consortium (FSTC), comprising of financial institutions, hardware and software companies, government agencies and industry associations, have an electronic check project in the area of electronic payments [FST]. FSTC electronic check system is a much more general-purpose system. It works muck like a paper check system in real life. But because of the necessity of using some kind of hardware devices, the usefulness of this system is limited.

So far, none of these existing electronic check systems has found acceptance by consumers or financial organizations. How to make the check-like systems more efficient and more practical, is a problem that all researchers, whether from universities or banking industry, are facing. The problem can be approached from two aspects: using more efficient cryptgraphic technologies to enhance the security and improve the efficiency; constructing more efficient communication tunnels between the electronic check systems and the existing financial organizations.

9.4.3 Electronic Cash Systems–Ecash

Normally, there are three or four parties involved in an electronic cash system: clients, merchants, banks and sometimes a trusted third party. The following main protocols are included in an electronic cash system: withdrawal protocol to withdraw money from banks; payment protocol to make a purchases; deposit protocol to get merchants credited. In some systems, there are also registration protocol and tracing protocol to realize some specific features.

According to whether the bank is involved in the payment protocol or not, when customers and merchants make the purchase, electronic cash systems can be further divided into two models: on-line and off-line.

Electronic cash systems are similar to the currencies used in the real world and this makes electronic cash systems more easy to be accepted by users. But unlike the currencies in our real life, electronic coins have a danger that they could be double spent, which means that a malicious user can make more than one purchase with the same electronic coin. Double spending resistance is a necessary requirement in electronic cash systems.

Many electronic cash systems have been proposed. The first off-line and double spending resistant electronic cash system was introduced in [Chaum90], and more efficient variants were designed in [Franklin93], [Brands93], [Brands94], [Ferguson94]. Systems in [Chaum90],[Brands93], [Brands94], [Chaum93] use wallets with guardians to prevent the systems from a double spending attack.

Among these systems, NetCash [Medvinsky93], Mondex [Mondex], CAFE [Boly94] and Ecash (DigiCash) [Chaum92] are the most well known. NetCash uses on-line checking to make the system double spending resistant. Mondex is an off-line electronic cash application built for MULTOS card operating system chip cards. A proprietary Chip-to-Chip protocol is used in Mondex to make secure transactions. The problem with Mondex is that all users should have a device to read chip cards attached to their machines. CAFE (Conditional Access For Europe) project was an European Community's program ESPRIT (project 7023). CAFE was designed to be a prepaid off-line electronic payment, where the original idea came from [Brands93], [Brands94].

Ecash was developed by DigiCash, a company specializing in electronic payment systems founded by David Chaum in 1990. It is a fully anonymous electronic cash system using the blind signature technique. RSA public-key cryptography technique is the basis of the system security. Anonymous electronic cash system was introduced by David Chaum [Chaum85], [Chaum90], [Chaum92]. David Chaum is a pioneer in the electronic payment area and sometimes has been called the father of digital cash. Unfortunately, his company, DigiCash went bankrupt in September 1998.

The goal of Ecash was to provide an anonymous, on-line digital cash system which was easy to implement and to use while being secure enough to make transfers over insecure networks. We will present Ecash in detail in the rest of this section.

Blind signature with RSA

Blind signature, first introduced by David Chaum ([Chaum83]), allows a person to get a message signed by another entity without revealing any information about the message to the signer. Blind signature is the most common cryptographic tool used in electronic cash systems to achieve system anonymity.

Blind signature with RSA works as follows: Suppose entity A has a message m which she wishes to be signed by entity B, and she does not want B to learn anything about m. Let (n, e) be B's public key and (d) be her private key. A generates a random value r such that $gcd(r, n) = 1$ and sends

$$m' = r^e \; m \; mod \; n$$

to B. The value m' is "blinded" by the random value r, and hence B can derive no useful information from it. B returns the signed value,

$$s' = (m')^d = (r^e m)^d \; mod \; n$$

to A. Since $s' = rm^d$ mod n, A can obtain the true signature s on m by computing

$$s = s' r^{-1} \; = m^d \; mod \; n$$

Now A's message has a signature which she could not obtain on her own. This signature scheme is secure provided that factoring and root extraction remain difficult. However, regardless of the status of these problems the signature scheme is unconditionally "blind" since r is random. The random r does not allow the signer to learn anything about the message even if the signer can solve the underlying hard problems.

Ecash architecture

There are three parties: bank, client and merchant, involved in

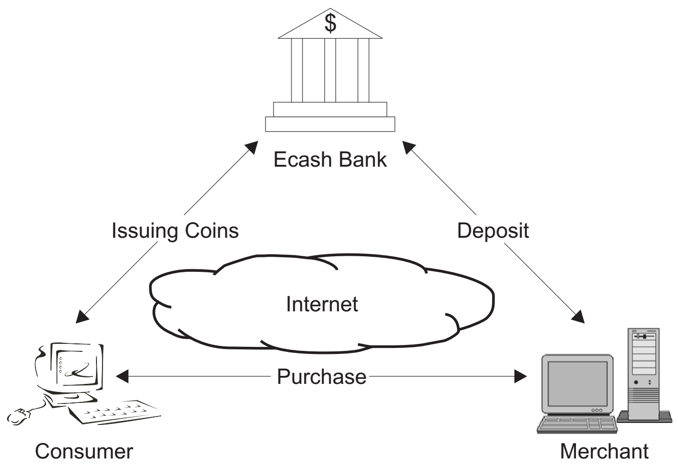


Fig. 9.4-17: Ecash architecture.

the system. Special client and merchant software are needed in Ecash as shown in Fig. 9.4-17.

Both client and merchant must have accounts at the same Ecash bank so that they can process a purchase. There is a client wallet, known as *cyberwallet*, to be installed in the client's machine to withdraw and store coins from the Ecash bank, and to process a purchase with merchants. The wallet keeps all transaction records and fulfills most of the tasks on behalf of the client. For some specific merchants or specific amounts, it can be configured to finish a purchase automatically.

Ecash coins

A coin in Ecash system is a unique serial number generated by a client's wallet and signed blindly by the Ecash bank. The serial numbers in Ecash systems should be large enough so that the chance that two different clients generate the same serial number is neglectable.

RSA-based blind signature, developed by David Chaum [Chaum85] was used as the blind signature in Ecash system so that the Ecash bank can not get any information on the serial numbers, making the system fully unlinkable and anonymous.

It should be pointed out that, in Ecash system, there are different signing keys (Key_Deno) for each coin denomination to assign various values to a coin. This is because the bank is unable to see any information on the coin including the value. The client has to inform the bank of the value it wants the blinded coin to be worth and then the bank can blindly sign the coin using the private key corresponding to that denomination.

An Ecash coin looks like in Fig. 9.4-18.

S_Num	Expiry_Date	Currency	Dey_Deno	$\{S\_Num\}_{SKKey\_Deno}$

Fig. 9.4-18: Coin format.

$\{S\_Num\}_{SK_{Key\_Deno}}$ is the result signed using the secret key of corresponding denomination of the bank.

Withdrawing coins

As shown in Fig. 9.4-19, to withdraw a coin from the bank, a client takes the following steps using an RSA-based blind signature scheme:

- Chooses a random blinding factor r and generates a serial number S_Num.
- Blinds theS_Num by multiplying it by the blinding factor raised to the power of the public exponent e : $S\_Num \cdot r^e(mod\ n)$. (n, e) is the public key of the corresponding denomination of the bank.

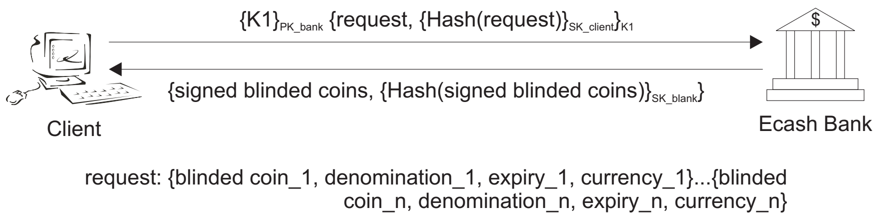

{K1}PK_bank {request, {Hash(request)}SK_client}K1
{signed blinded coins, {Hash(signed blinded coins)}SK_blank}
Client
$
Ecash Bank
request: {blinded coin_1, denomination_1, expiry_1, currency_1}...{blinded coin_n, denomination_n, expiry_n, currency_n}

Fig. 9.4-19: Withdraw coins.

- The bank signs the blinded S_Num using the corresponding secret key d and returns it to the client: $(S\_Num \cdot r^e)^d(mod\ n) = (S\_Num)^d \cdot r(mod\ n)$.
- The client gets the signed Ecash coin by dividing out the blinding factor: $(S\_Num)^d \cdot r/r(mod\ n)$.

Many coins of different denominations can be withdrawn in a single request. The request must be signed with the client's secret key, and the whole request is protected by encrypting it with the bank's public key (PK_bank). As we can see from the Fig. 9.4-19, the digital envelope technique is used here again. The withdrawal request contains some information including unsigned blinded coins, indication of the required denominations, expiry and currencies.

The blindly signed coins are returned to the client without encryption because only the client who generated the random blinding factor can unblind the coins and later spend them.

Payment : spending coins

Once a client has withdrawn coins from the bank and stored them in her wallet, she can spend these signed coins at a merchants site. As shown in Fig. 9.4-20, a complete purchase can be as follows:

- Client selects items from a merchant's Web site and sends an *order* to the merchant. This could be done by submitting a form to the merchant's web site.
- After receiving the order, the merchant sends a *payment request* back to the client to invoke the client's wallet. The request contains details about the purchase information: the amount, the currency, the time, the merchant's bank, the merchant's account ID and a description of the items selected. The request is sent to the client without encryption.

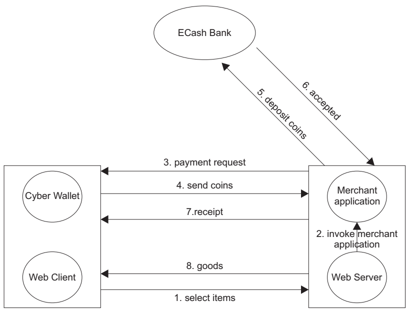


Fig. 9.4-20: Payment in Ecash.

- The client then decides (or the wallet was configured to decide automatically) whether to make the payment. Once the client decides to pay, a *payment* will be sent to the merchant to process the purchase. A payment instruction consists of the payment information and encrypted coins:

$$\begin{aligned} payment &= \{payment\_Info, \{coins, H(payment\_Info)\}_{PK\_bank}\} \\ payment\_Info &= \{BankID, amount, currency, time, expiry, merchant\_ID, \\ &\quad numberofcoins, H(description), H(payer\_code)\}. \end{aligned}$$

A hash of the order description, $H(description)$, is included within the payment information. Because the merchant knows the order, the merchant can compute a new hash using the order copy from her side and compare it with the $H(description)$. If these two hashes are the same then the merchant can be sure that the client agrees on this order. *Payment_code* is a secret generated by the client to prove to the bank, after the merchant has deposited the coins, that she made the payment. A hash value of the secret, $H(payer\_code)$ is included in the payment information so that the bank can make sure that it was generated by the client when she reveals the original $payer\_code$ value.

The coins used in the payment are protected by being encrypted using the public key of the bank(PK_bank) so that nobody else except the bank can read the coins. Also a hash of *payment_Info*, $H(payment\_Info)$, is encrypted to protect all messages from being altered.

Deposit coins

The merchant can get credited through the deposit process. At this point, a normal purchase with Ecash coins will be completed.

- Upon receiving the payment message, the merchant compares hashes to make certain that the client agreed on the order and then sends a *deposit request* to the bank including the payment message as part of the content. This message can be optionally signed or encrypted before being sent to the bank.
- The bank verifies the signature and checks the coins contained in the payment message to make sure they have not been spent, credits the merchant's account and returns a receipt to the merchant.
- After receiving the receipt, the merchant delivers the items to the client with the receipt from the bank. In case the merchant rejects to deliver the items to the client, the client can send the value *payer_code* to the bank to prove that she has made the payment.

Double spending

Double spending means to spend a coin more than once. This is fatal to electronic cash systems and must be prevented. Ecash system is designed to be double spending resistant.

Since the bank in Ecash system can not know the serial numbers on the coins it issues, the bank can not record these serial numbers to prevent double spending. So it is difficult to prevent double spending in the electronic cash systems providing full anonymity. In Ecash, the bank will store all coins deposited back to the bank and check on-line to make sure a coin is not spent twice. A valid unspent coin should satisfy the following conditions:

- Have a valid signature from the bank;
- Not appear in the deposited coins database;
- Be still not expired.

Although the bank can delete those coins which have been spent after the expiry date, a large database has to be maintained by the minting coins bank. Furthermore, Ecash can not prevent such misuses as blackmailing [Solms92] from happening.

Remarks

Ecash is an on-line electronic cash system providing secure, fully anonymous coins over insecure open networks. It is suitable for small payments without the need for applying a credit card. On the other hand, a wallet software must be installed at the client's machine to withdraw, store and spend coins from the bank. Furthermore, the computationally intensive cryptography, maintaining and searching a large database of all spent serial numbers in order to be double spending resistant, prevent the Ecash system from being used widely in the electronic payment area.

Several systems have been presented in [Traoré99], [Qiu02] providing revokable anonymity integrating a cryptological technique called group signature [Chaum91]. Under special circumstance, the anonymity of an electronic coin can be removed to reveal the identity of the coin. There is a trusted thirty party involved in these systems. Some other systems without any trustees are also introduced in [Sander99], [Kügler01], [Pfitzmann00].

9.4.4 Micropayment Systems–Millicent

With the rising necessity of intangible goods (e.g. information, CDs, games) in worldwide economies over the Internet and their instantaneous delivery at negligible cost, micropayment systems play an important role in billing for products and services for small amounts of money. The goal of micropayment systems is to keep the cost of transactions with small amounts of money low.

Unlike previous systems(e.g. iKP, SET, NetBill, Mondex) using public key cryptography, micropayment systems use efficient one-way, collision resistant hash functions or only private key cryptography which minimize the computational and storage costs.

Micropayment systems are apparently not as secure as the previous electronic payment systems. However, if a micropayment system is designed so that a customer is only at the risk of losing a few cents, and the cost of counterfeiting a coin is computationally higher than the value of the coin, then the security of the micropayment system is considered to be adequate. This is the basic security assumption of micropayment systems.

Millicent [Glassman95] is a micropayment system developed by DIGITAL Equipment Corporation. The goal of Millicent is to allow transactions of small amounts while keeping the system secure.

Millicent uses an electronic currency called *scrip* to make the system unforgeable, unusable if copied and inexpensive to create and validate. A *scrip* is related to a specific vendor in that it is valid or has a value only at one particular vendor.

MilliCent Model

Fig. 9.4-21 show that there are three entities in the Millicent system: *brokers*, *vendors*, *customers*. Brokers are assumed to be the most trustworthy.

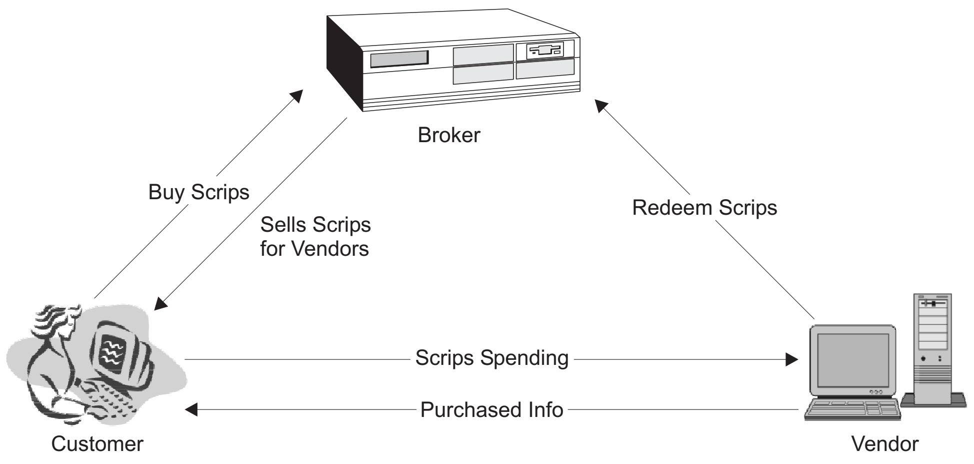


Fig. 9.4-21: The Millicent model.

Brokers serve as accounting intermediaries between customers and vendors providing all the different vendor *scrip*s for customers. In other words, the brokers buy *scrip*s from different vendors and sell them to the customers. The Millicent vendors, who sell low value services or information to customers, accept *scrip* issued by themselves. The customers buy *scrip* from the brokers and then can make purchase at the specific vendor.

Millicent *scrip*

scrip is the currency used within the Millicent system. The main properties of *scrip* listed in [Glassman95] are as follows:

- It is vendor-specific and only has value at a specific vendor.
- It can only be used by its owner.
- It is tamper resistant and hard to forge or change its value.
- *scrip* can be efficiently produced, validated without use of public key cryptography and is protected using a secure hash function.
- It can be spent only once.
- It has a visible serial number which is used to prevent double spending.

As shown in Fig. 9.4-22, A *scrip* structure contains the following fields:

- *Vendor* This field identifies the vendor for the *scrip*.
- *Value* The amount of the *scrip*.
- *ID#* The unique identifier of the *scrip*. It is used to prevent the *scrip* from double spending and some portion of it is used to select the *master_scrip_secret* used for certificates.
- *Cust_ID#* An identifier used to produce *customer_secret* which is used to protect the *scrip* and it must be unique to every customer.

- Expiry. The expiration time for the *scrip*t. Used to reduce the system's memory size.
- Extra info. The info is extra data describing the customer to the vendor and might be age, country of residence, etc. It is helpful to the vendor to make a sales decision.
- Certificate. It is a kind of signature providing authentication. It is not based on public key algorithms. Instead, a hash function is used.

There are three secrets, *customer_secret*, *master_customer_secret* and *master_scrip_secret*, existing in the Millicent system. Under the secure network connection case, the *customer_secret* is used as the sharing key to encrypt the network connection using a symmetric or private key encryption algorithm. *Master_customer_secret* is used to produce *customer_secret* and *master_scrip_secret* is used to create the signature of the *scrip*, called *certificate*. The secrets are all used in a way that shows knowing the secret but does not reveal any contents of the secret.

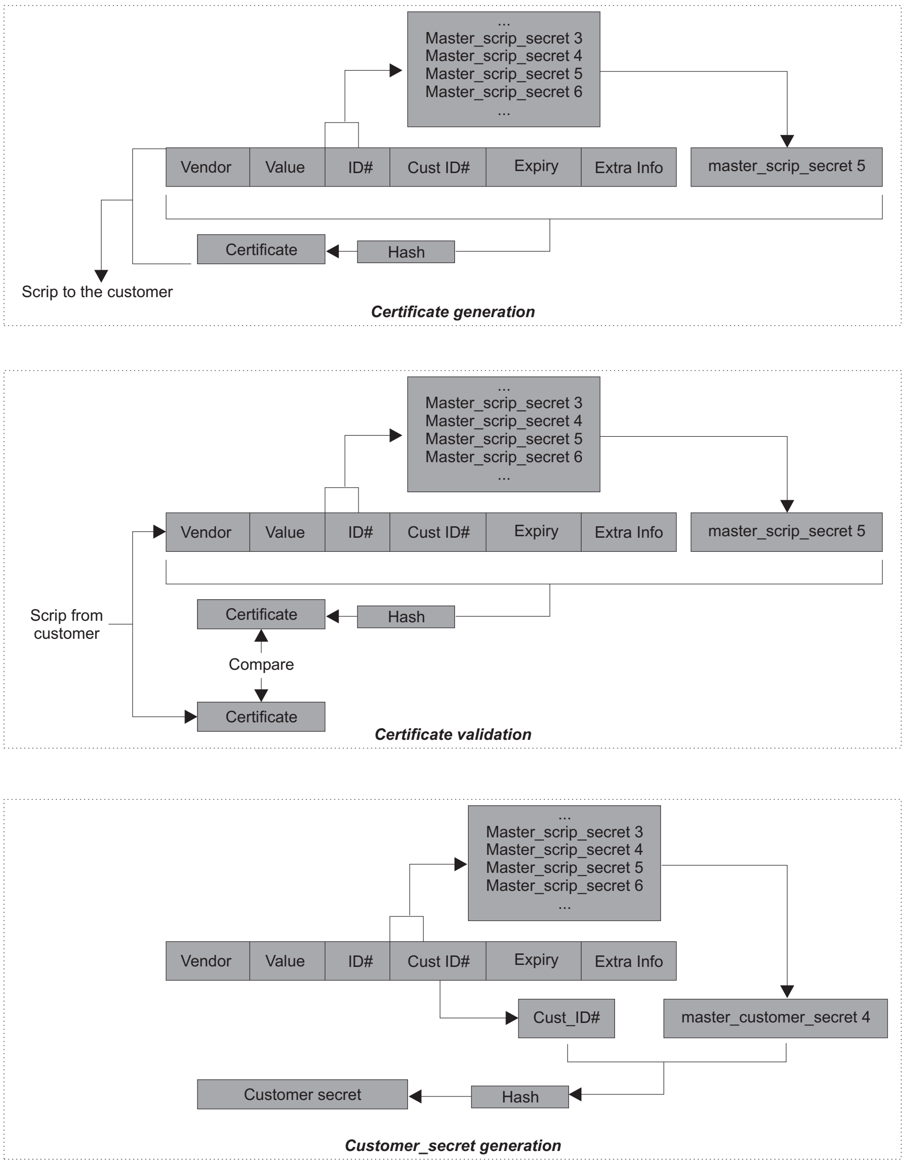


Fig. 9.4-22: The generation and validation of the certificate and generation of customer secret.

Generation and validation of *scrip* certificate

The certificate generation and validation of a *scrip* are shown in Fig. 9.4-22.

The vendor maintains a list of *master_scrip_secret*s, numbered from 1 to N which will be used to prevent *scrip* from being forged. Only the vendor who mints the *scrip* knows these secrets. When minting a *scrip*, the vendor selects a *master_scrip_secret* according to some parts of $ID\#$, for example the highest three bits in the $ID\#$, and then a secure one-way hash function (e.g. MD5, SHA1) is used to generate the certificate so that the certificate can prevent *scrip* from being altered.

When validation is going on, the vendor chooses the same *master_scrip_secret* using the same part of the $ID\#$, generates again the certificate and compares it with the certificate contained in the *scrip* to authenticate the *scrip*. Also during validation, the vendor checks the list of *scrip* $ID\#$ to make sure it is not a double spending *scrip*.

The Millicent protocols

Scrip is the basis of a Millicent protocol. There are three different protocols to send scrip over the network:

- *Scrip in clear.* It is the simplest protocol in which the customer sends the *Scrip* unprotected in plaintext over the network. Also the purchased contents and *Scrip* change information returned from the vendor is in cleartext.

- *Private and Secure.* In this case, Millicent can set up a secure tunnel between the vendor and her customers using a *customer_secret* as a sharing key with any symmetrical encryption algorithm (e.g. DES, IDEA etc.).

  Fig. 9.4-22 shows how to generate the *customer_secret* when the scrip is created. Some portion of the *Cust_ID#* is used as an index to select a *master_customer_secret*. The *master_customer_secret* list is maintained by the vendor and only known to the vendor. Then the *customer_secret* can be generated by hashing the *Cust_ID#* with the selected *master_customer_secret*. On receiving the *scrip*, the merchant can recalculate the *customer_secret*. The customer must obtain the *customer_secret* from a broker.

- *Secure without encryption.* A fully encrypted network connection may be too expensive or an overkill for a micropayment system. This third variant protocol uses the *customer_secret* as one part of the message with the *Scrip* and purchase request to generate a digital signature using a secure hash function, just as the generation of the certificate in the scrip. It provides secure authentication without any encryption.

Millicent is an efficient, flexible and simple micropayment system, in which no public key encryption is used and verification is decentralized. However, *scrip* buyers can be spoofed since only the issuer of scrip can verify *scrip*.

PayWord [Rivest96] and MicroMint [Rivest96] are both typical micropayment systems. The PayWord protocol is a credit-based protocol proposed by Ron Rivest and Adi Shamir in 1996. It is based on a chain of hash values called *payword*. A payword chain is vendor-specific and the user authenticates a complete *payword* chain to the vendor with only a single public key signature. Each *payword* represents a particular unit of value. Until so far, there are still many researchers concerned about the security and the practicality of PayWord [Adachi03], [Yang03]. MicroMint is the second micropayment scheme proposed by Ron Rivest and Adi Shamir who also designed PayWord in the same paper. MicroMint gives up completely using public key cryptography. It is debit-based to customers and brokers, while it is credit-based to a vendor. MicroMint adapted brokers to mint electronic coins which can be used at any vendor without an on-line verification.

An important feature of micropayment systems is that they use simple secure techniques to reduce the communication and speed up the transaction process. Although there is still no large scale use of these systems, micropayment systems still provide an ideal way to meet the demand for efficient and secure small-value payment systems over insecure networks. As more and more new techniques are being deployed and more schemes coming up, micropayment systems will bring much more commercial chances and profit to the whole open network community.

10 Security Aspects in Mobile Agent Systems

10.1 An Overview of Agent Technology and the Mobile Agent Paradigm

This chapter gives an overview of the agent technology and the mobile agent paradigm. An evolution path of mobile agent paradigm is illustrated. A short description of advantages and disadvantages of the mobile agent paradigm, a survey of mobile agent systems and research areas in mobile agent have been provided. In the next section, the security aspects in a mobile agent system will be discussed in detailed.

10.1.1 Overview of agent technology

At present, the computing environment migrates from individual standalone systems to open systems. Due to the properties of open and complex system, system components must be able to response towards changes dynamically. In other words, there is a need for some degree of autonomy. Agent-based computing has emerged to deal with the above scenario. Before we present a set of agent-based applications, we have to describe what we mean by such terms as ‘agent’ and ‘agent-based systems’. Indeed, there are different labels of agents ranging from the generic *autonomous agent, software agents, and intelligent agents* to the more specific *interface agents, mobile agents*, and so on. The usage of the agent metaphor that leads to such different uses of the term is both strength and weakness. Its strength is that it can be applied in many applications. The weakness is that the term agent is used so often that there is no commonly accepted notion of what it is that constitutes an agent. We conclude that there is no agreement on what is the standard definition of an agent, therefore, for many researchers their only feasible solution is to provide their own definition. According to [Luck03], agents can be defined to be autonomous, problem-solving computational entities capable of effective operation in dynamic and open environments. From [Jennings98], an autonomous agent is an agent, which can act without the direct intervention of humans and should have control over its own action and internal states. Agents are often deployed in environments in which they interact, and may cooperate, with other agents that have possibly conflicting aims. These environments are known as multi-agent system. From [Luck00], we have the definition of a multi-agent system as a distributed system in which several distinct components, each of which is an independent problem-solving agent come together to form some coherent whole. There are several applications to which agent technology can be adapted.

Agent-based applications

Classified according to the application domain (more details in [Jennings98]) agent technology can be utilised in industrial applications (e.g. process control, manufacturing and air traffic control), commercial applications (e.g. information management, e-commerce and business process management), and medical applications (e.g. patient monitoring). All of the application domains use software agents as a key design. From [Nwana96], Nwana presents a topology of software agents. They propose that agents exist in a truly multi-dimensional space. However, for the sake of clarity of understanding, they identify seven types of agents according to mobility, autonomy, learning (i.e. it reacts and/or interacts with its external environment; such that its performance increases) and cooperation (i.e. it cooperates with other agents) namely; Collaborative agents, Interface agents, Mobile agents, Information/Internet agents, Reactive agents, Hybrid agents and Smart agents.

10.1.2 The Mobile Agent Paradigm

The mobile agent paradigm is the overlapping concept between code mobility (i.e. mobile code) and the agent technology. In [Carzaniga97], Carzaniga et.al. explain code mobility as the capability to dynamically change the bindings between code fragments and the location where they are executed. Code mobility is not a completely new concept. It was first introduced in the period where a single high performance mainframe was utilized as the central computation entity, e.g. remote batch job submission and the use of postscript to control printers. Code mobility can be viewed as one of the variance of process migration implementation. In [Milojicic99a], Milojicic et. al. define process migration as the act of transferring a process (i.e. a process is an operating system abstraction representing an instance of a running computer program) between two machines during its execution. The purposes of process migration are relatively tied with the type of applications that use migration. For instance, when applied to load distribution, its goals are to access more processing power or, when utilized in mobile computing, to migrate running applications from a host to their mobile computer as they connect to a network at their current location or back again when they disconnect. From [Fuggetta98], Fuggetta et. al. illustrate a basic foundation of code mobility deployment and the set of code mobility design paradigms (i.e. remote evaluation, code on demand and mobile agent). The following figure, found in [Karnik98], presents an evolutionary path from the client/server paradigm to the mobile agent paradigm. To discuss the differences in design paradigm, initially, we define some common terms as follows: a host is an execution platform, a code is a program to be executed, and resources can be generally interpreted as memory, CPU, or data contained in a host.

- **Client-Server Paradigm:** This paradigm is well known and broadly used. In it, host B offers a set of services. Resources and codes are hosted by host B. Host A requests the execution of a service at host B. Host B performs the requested service by executing the corresponding code and accessing the involved resources co-located at B. At the end, host B returns the result of code execution to host A.

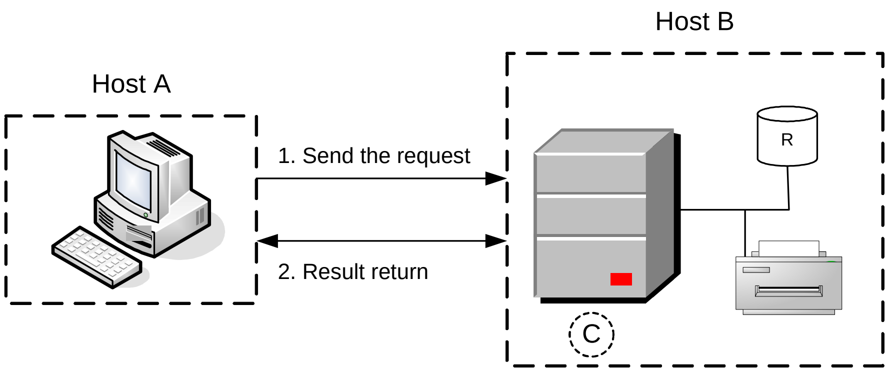


Fig. 10.1-1: Client Server

- **Remote Evaluation:** Host A has the code but lacks the necessary resources located at host B. Host A sends the code to host B. Subsequently, host B executes the code using the resources available there and sends the result of computation back to host A.

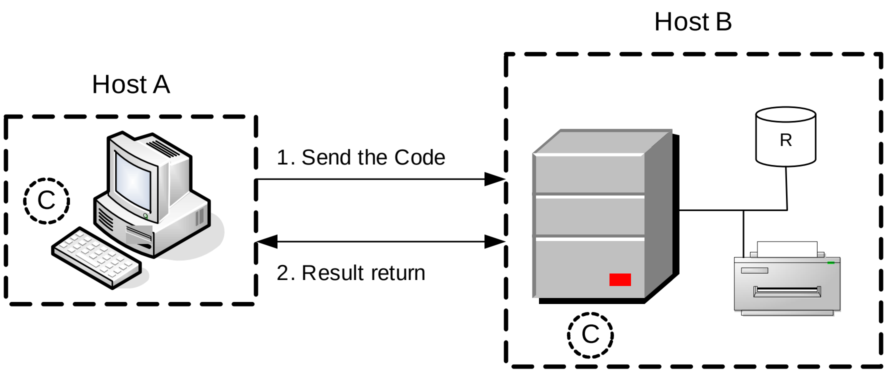


Fig. 10.1-2: Remote Evaluation

- **Code on Demand:** Host A has the resources; however, it does not have a required code for manipulating resources. It requests host B for the code. After receiving the code, host A executes the code.

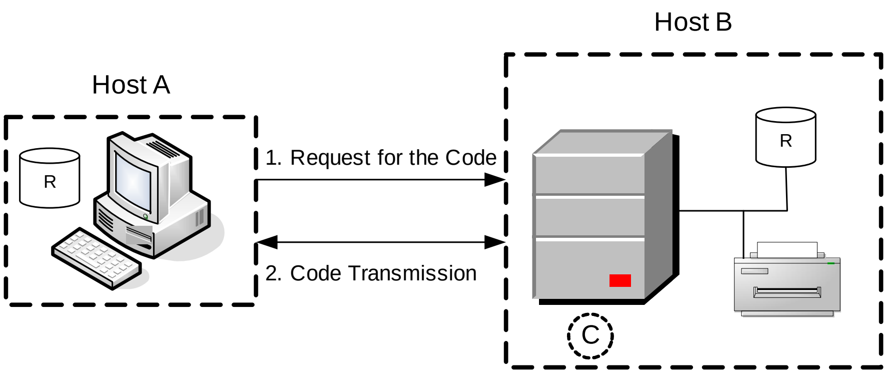


Fig. 10.1-3: Code on Demand

- **Mobile Agent:** Host A has the code in hand, instead of forwarding/requesting services offered by host B. This time however, the code itself is able to migrate to the new host and interact locally with resources to perform its tasks. This method is quite different to the previous three examples, in this instance an entire component is migrating, along with its associated data and code.

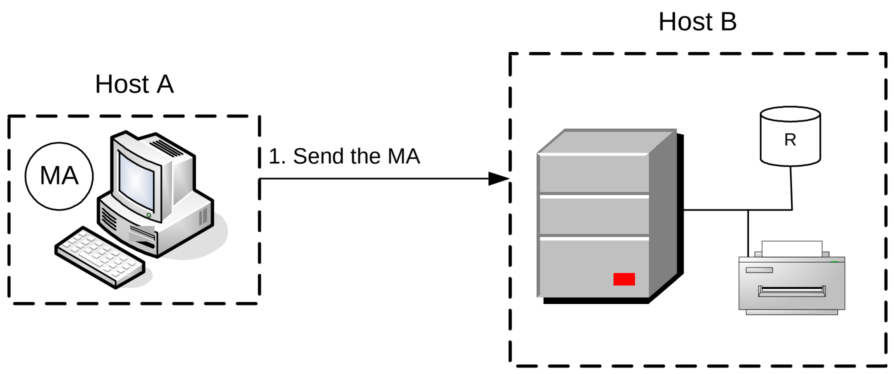


Fig. 10.1-4: Mobile Agent

10.1.3 Advantages and Disadvantages of the Mobile Agent Paradigm

With the unique ability to transmit itself from one host in a network to another host and autonomous capability in performing tasks on behalf of its owner, the mobile agent paradigm grants some benefits over the traditional distributed computing approaches. According to [Lange99], the mobile agent paradigm offers the following benefits:

- **Network load reduction:** Traditional communication protocols (e.g. challenge responses, client-server) involve multiple interactions to complete a target task. Applications dealing with massive information transfer (e.g. information retrieval and filtering) in a low/high bandwidth connection (i.e. in case of a low bandwidth connection induces significant network load, the effect of massive data transfer can obviously be noticed). As In [Das03], mobile agents have the potential to substantially improve the speed and efficiency with which distributed and heterogeneous data is retrieved. By moving the computation to the data, retrieval times can be reduced by the elimination of unnecessary data transfer.

- **Network latency minimization:** A real time system (e.g. network monitoring, remote controlling, and network management) requires a fast response towards dynamic changes of system environment. Communication delay in a real time system causes slow response (e.g. when remote procedure call is used for a remote countermeasure activation) towards reported problems that leads to, in a worst case, a system breakdown. Conventional network management is based on SNMP (i.e. Simple Network Management Protocol) and often run in a centralized manner. Although the centralized management approach gives network administrators a flexibility of managing the whole network from a single place, it is prone to information bottleneck and excessive processing load at the manager and heavy usage of network bandwidth. The mobile agent paradigm offers a better solution by dispatching a mobile agent to operate locally on remote systems. A mobile agent can monitor and execute autonomously when problems occur.

- **Providing asynchronous, dynamic and autonomous computation:** Mobile agents are capable of reacting autonomously to changes in the system. They can cooperate with other agents in the system to achieve their goals.

- **Robustness and fault tolerance:** Mobile agents' ability to react dynamically to unfavorable situations and events makes it easier to build robust and fault-tolerant distributed systems. If a host is to be shut down, all agents executing on that machine are warned and given time to dispatch and continue their operation on another host in the network.

From the list above, the mobile agent paradigm provides appealing and intuitive benefits. Unfortunately there are many difficult problems that must be solved in order to make mobile agent-based applications work properly. From [Vigna04] and [Milojicic99b], we gather the reasons of failure when utilizing the mobile agent paradigm as follows:

- **Difficulties in a mobile agent system design:** A mobile agent system design including testing and developing is rather complex due to system components interoperation and the distribution of the heterogeneous host platforms in a network.

- **Difficulties in security aspects realization:** As its major vulnerability, security aspects in the mobile agent paradigm can be broadly grouped into two groups namely: protecting host from malicious mobile agents and protecting mobile agent from malicious hosts. Further details will be presented in Section 10.2.

- **Difficulties in communication infrastructure design:** A communication infrastructure must be used when a mobile agent communicates with other agents (which reside on the same or different platforms) or platforms. To make communication possible the following must be achieved. Each interaction requires that the format used to exchange data and the meaning that is associated with the data is understood and agreed upon by both the agent and the partner of the interaction. When a mobile agent wants to communicate by message passing with another agent on a different host, name and location of the target agent must be known. Every host, which can possibly be the recipient of mobile agents, must have a common communication infrastructure.

- **No killer application:** The mobile agent paradigm can be used to implement applications in several cases, but these applications can also be implemented by traditional techniques.
- **Similar to worms:** A mobile agent autonomously activates the transfer of its code and data to another visited host where it resumes its execution. This feature can be ambiguously interpreted as worm-like attacks.

10.1.4 Survey of Mobile Agent Systems

The mobile agent systems can be grouped according to their development purpose into two groups namely: commercial and non-commercial. Some examples of commercial mobile agent systems are Telescript/Odyssey, Voyager, Concordia, Javaspaces, Crystaliz, Kyma Jumping Bean. On the other hand, some samples of non-commercial mobile agent systems are D'Agents, ARA, Mole, Tracy, and NOMADS. There are also open source agent systems written in java available such as Aglet, Bee-gent, BOND, Cougaar, JADE and Tucson. The state of the art in frameworks for mobile agent programming is almost exclusively java. However, there are some exception such as D'Agent (developed in TCL),Ara (that supports TCL and C/C++). According to [Mahmoud04], Mahmoud explains that mobile agent programming can be written in any language (e.g. Perl, TCL, Python, etc.) and may be executed in either machine or interpreted language. Nevertheless, in order to support heterogeneity it is preferable and beneficial to program mobile agents in an interpreted language. The Java language has a number of advantages that make it particularly appropriate for Mobile Agent technology. While Java is by no means the only language being employed the Mobile Agents, it is arguably the best choice. The reasons for this are:

- **Portability**. Its use of bytecodes and its interpreted execution environment mean that any system with sufficient resources can host Java programs. There are even machines being built today that execute Java natively. For Mobile Agents this is a tremendous opportunity. The more platforms capable of executing the agents' code, the better.
- A second advantage comes from **the ubiquitous nature of Java on the Internet**. Because it is embedded in many Web browsers, as well as application servers, there are many platforms deployed already. Application Programming Interfaces such as AWT, the Advanced Windowing Toolkit, JFC, the Java Foundation Classes, and JDBC, Java Data Base Connectivity, are leading toward even more deployment of Java. Additionally, this deployment exactly targets the sort of services that agents can best use.

- Another major advantage is **the proliferation of tools that support Java programmers**. Many programmers are already familiar with C++, which Java resembles in many ways. Added to that is the migration of existing tools to Java and the creation of many more. The net result is an abundance of high quality, easy to use tools for both development and debugging.
- Finally, there is **the movement of major segments of the software industry to Java**. Not only will Java be here for many years to come, it will be employed in ever increasing applications.

10.1.5 Research in the Area of Mobile Agent

In the last decade, research and development of mobile agents are growing very fast. However, many research problems are still not solved completely. Many projects conducted so far had the goal to prove the possibility to implement mobile agents in different programming languages and showed that it is worth using mobile agents for particular application domains. In [Milojicic99c], Milojicic et. al. conclude that there are four research directions for the mobile agent paradigm namely: application, infrastructure, security and standard.

- **Application:** According to [Lange99], there are a lot of conceivable applications for mobile agent or mobile code namely:
  - Electronic commerce: E-Commerce is increasingly assuming a pivotal role in many organisations. It offers opportunities to significantly improve (make faster, cheaper, more personalised and/or more agile) the way that businesses interact with both their customers and their suppliers. E-Commerce, as new opportunities for business making, can use the mobile agent paradigm in order to efficiently utilize its potential. For example, integrating a mobile agent paradigm to e-Commerce in a trading scenario can be viewed as vendor agents displaying goods and services and buying agents (i.e. represent buyers) making negotiations, and finally purchasing on behalf of buyers [Guttman98].
  - Personal assistance: In case of roaming devices such as Personal Data Assistants (PDAs) or laptops, they are often disconnected from a permanent network for the sakes of cost of network connection and power consumption in message communication. This gives an opportunity to use mobile agents by a device dispatching its mobile agents to perform tasks on its behalf.
  - Information retrieval: Within the traditional client-server paradigm, one is likely to involve massive data transferring across the networks which induces high bandwidth consumption and time delay. Utilizing a mobile agent paradigm can reduce network load and provide data searching locally.
  - Real time monitoring: Mobile agents can work locally on remote sites. An agent can monitor a given information source without being dependent on the system from which it originates.

  - Parallel processing: More than one mobile agent can be dispatched to different sites to perform tasks in parallel.
  - Web services: These have been proposed as an important paradigm for supporting heterogeneous distributed computing by focusing on a core group of Internet standards such as XML, SOAP and WSDL, in the development of web-centric applications, particularly for B2B transactions. Web services define a technique for describing software components to be accessed, methods for accessing these components and methods that enable the identification of relevant services provided
  - Intrusion detection: In [Kruegel01], Kruegel et. al. discuss advantages and disadvantages when applying the mobile agent paradigm to a distributed intrusion detection scheme. They conclude that only a few systems adopt mobile agents to perform security related tasks, even though its advantages seem to be impressive. This stems from the fact that the benefits are not introduced automatically and often the disadvantages outweigh the intended improvements.

  The list is not exhaustive, but it names some applications where mobile agents can be utilized.

- **Infrastructure:** This can be viewed as naming, locating, controlling and communicating.
- **Security:** This can be categorized mainly into two important classes namely: protecting a host from malicious mobile agents and protecting a mobile agent from malicious hosts.
- **Standards:** The systems, from Section 10.1.4 differ widely in architecture and implementation, thereby impeding interoperability and rapid deployment of mobile agent technology in the marketplace. To promote interoperability, some aspects of mobile agent technology need to be standardized. Currently there are two standards for mobile agent technology: The Object Management Group's Mobile Agent System Interoperability Facility (MASIF) [Breugst98] and the specifications promulgated by the Foundation for Intelligent Physical Agents (FIPA). MASIF is based on agent platforms and it enables agents to migrate from one platform to another. FIPA spec is based on remote communication services. The former is primarily based on mobile agents travelling amongst agent systems via CORBA interfaces and does not address inter-agent communication. The latter focuses on intelligent agent communication via content languages and does not say much about mobility. FIPA adopts an agent communication paradigm, which can better express the nature of cooperation and is more suitable for integration with other AI technologies. MASIF adopts a mobile agent paradigm, which is more appropriate in situations where dynamic and autonomous swapping, replacement, modification, and updating of application components are required. FIPA 2002 provides the possibility of MASIF/FIPA integration.

10.2 Security in a Mobile Agent System

The objective of this section is to demonstrate the security threats and existent countermeasures found in mobile agent systems. Section 10.2.2 presents the classification of security threats in a mobile agent system. Section 10.2.3 discusses some host protection techniques. Section 10.2.4 gives a brief survey of methods of protecting a mobile agent from malicious hosts' attacks.

10.2.1 Introduction

As one of the active research areas of the mobile agent paradigm, security aspects have received a lot of attentions from both research communities and academia. In [Montanari01], fully-scale adoption of mobile agent paradigm in untrustworthy network environments, such as Internet, had been delayed by several security complexities. Before proceeding further, some definitions are given as follows. A host platform is referred to as a computation environment that provides mobile agents to execute their tasks, to access system resources and to guarantee integrity and protection of mobile agents and its environment. We use 'host (agent)´ and 'host platform (mobile agent)´ interchangeably. A host also accommodates migration, naming and communication services. A mobile agent is a program (containing variables and code) that autonomously migrates from host to host to act on behalf of its owner in a heterogeneous network.

10.2.2 Classification of Security Threats

From [Janzen00], there are four classes of security threats (where the classifying criteria are based on the source of attacks and the victim).

Mobile agent against host (Host security)

This type of threat can be viewed as a mobile agent commencing malicious actions on the host's resources which it has gained access to. The host's resources can be conventionally illustrated as its computation resource (e.g. memory and CPU), stored data, and communication devices. Malicious actions can be categorized into two situations when the first is a mobile agent with an authorized access and the second is a mobile agent without an authorized access (see Fig. 10.2-1).

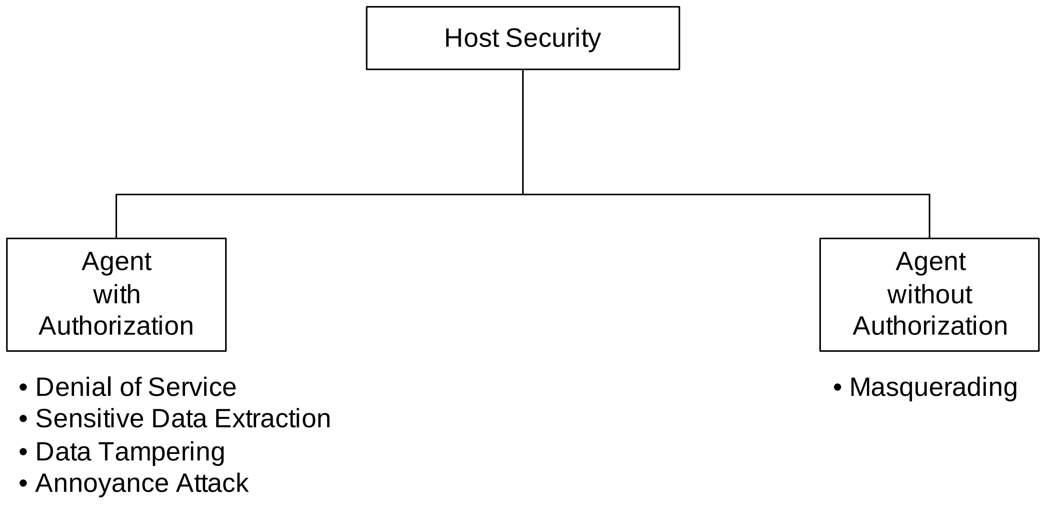


Fig. 10.2-1: Attacks against Host security

Malicious actions related to the first situation are shown to be:

- **Denial of Service:** A malicious mobile agent may attempt to consume the host's resources, which leads to a high processing time. In the worst case, a mobile agent can shut down the host.

- **Sensitive data extraction:** A malicious mobile agent uses its authorized access to extract some private data.

- **Data tampering:** A malicious mobile agent attempts to modify host data by deletions addition or alteration.

- **Annoyance attack:** A malicious host repeats some useless action, e.g. popup many new windows.

In the second situation, a mobile agent performs the following action to obtain an authorized access and then executes the same sort of attacks perceived in the first situation.

- **Masquerading:** A mobile agent masks itself as an authorized mobile agent to perform some attacks shown in the first situation. This type of attack is equivalent to a virus or a worm's invasion.

The above represent only a partial list of possible attacks. These vulnerabilities are stemmed from lacking of an efficient control protocol. An efficient control protocol should comprise of four steps namely an authentication scheme, a mobile agent's code investigation, a resource provision, and a mobile agent's activity monitoring. Please note that step 2 and step 3 can be rearranged (i.e. may be step 3 can be executed before step 2) or combined together. The definition of each step and a number of host protection methods will be further discussed in section 1.2.3.

Host against mobile agent (Mobile agent security)

This category has obtained a lot of attention from many research communities. Here it is usually the case that a host, which has full control over the environment, launches different types of attacks on a mobile agent currently residing on its site. There are many researchers who attempt to distinguish different types of attacks imposed on a mobile agent by a malicious host. For instance, [Janzen00] Janzen et. al. categorize attacks into four classes namely disclosure of information, denial of service, corruption of information and interference. Beirman et. al. [Beirman02], categorize their classification of malicious host security attacks into four classes namely integrity attack, availability attack, confidentiality attack, and authentication attack. Concluding from literature relating to classification of malicious hosts' attacks, generally, a malicious host can attack in two aspects namely; on mobile agent's components (agent's components comprise of its data (e.g. collected data or result of computation), variables (i.e. state variable and input variable), and code) and services provided to a mobile agent (see Fig. 10.2-2).

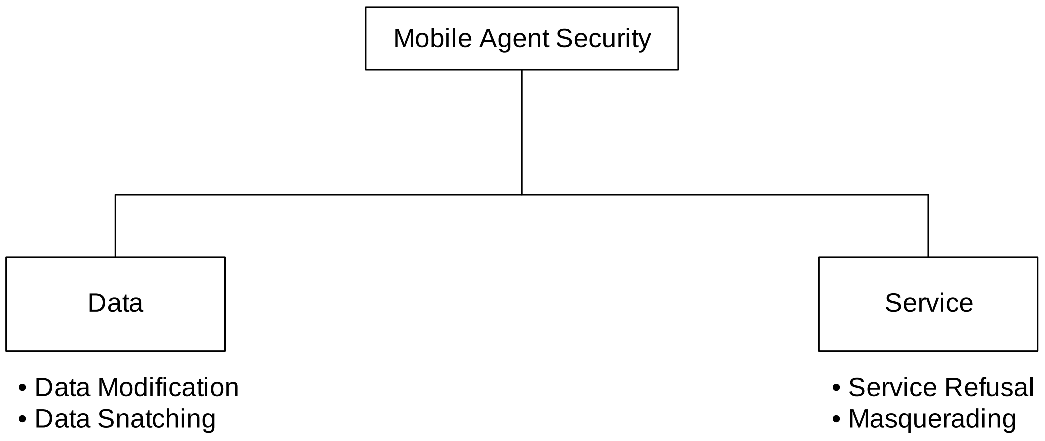


Fig. 10.2-2: Attacks against Mobile Agent Security

The first are mobile agent's components tampering and snatching. Tampering case can be explained as follows:

- **Modification:** We provide two general cases: namely code modification and data modification. In case of code modification, a malicious host might delete some part of the agent's code so that a mobile agent cannot work according to how it was implemented. In the worst case of code modification, a malicious host simply deletes a mobile agent out of his platform. Generally, a mobile agent must migrate to certain host platforms to collect some data. In case no precautions are taken, a malicious host is able to see the collected data. It can also modify (i.e. delete, alter) the collected data according to its goal.

In snatching case, a malicious host can perform as follows:

- **Eavesdropping:** In this attack [Alfalayleh04], a malicious host extracts the sensitive data of a mobile agent or intercepts intercommunication between mobile agents. Even though, there is no alterations this extracted information might be beneficial to a malicious host.

In respect to services, a malicious host can do the following.

- **Service refusal:** It is an action, which attempts to obstruct a mobile agent to accomplish its tasks. A well-known premeditative action is the denial of service. Denial of service can be accomplished in different ways. For example, a malicious host does not provide its communication device when a mobile agent needs to transfer a message to its interaction partner or postpone the requested service so that a mobile agent must reside in its environment for a long period of time.
- **Masquerading:** A malicious host hides its real identity and claims identity of another host in order to gain access to a mobile agent's data.

The above represent only a fraction of possible attacks. To resolve these vulnerabilities of a mobile agent emanated by malicious hosts, a set of counteractions must be available in each mobile agent system. A brief survey of existent counteractions will be further discussed in Section 10.2.4.

Mobile agent against other agents

This category represents a set of attacks in which a malicious mobile agent exploits security weaknesses of a target agent and a host to launch attacks against the target agent. A set of attacks can be implied as data tampering and denial of service. One form of data tampering is when a malicious agent overwrites or alters a target agent's code. A possible denial of service attack is for example when a malicious agent performs repeated activities with a target agent so that the target agent cannot continue its task. To handle this situation, a host must provide an efficient access control and an activity observation (i.e. monitor the behavior of a mobile agent towards other agents).

Other external entities against mobile agent system

This category represents a set of attacks in which external entities (this refers to all entities which are not included in a particular mobile agent system), threaten the security of a mobile agent platform (i.e. all mobile agents and all hosts constituting a particular mobile agent system). This set of attacks is roughly designated as masquerading (i.e. an external entity attempts to masquerade to be a mobile agent or a host which is legitimated), and communication attack (i.e. an external entity eavesdrops, intercepts or tampers messages sent between legitimate entities).

10.2.3 A Survey of Host Protection Methods

Now that we have looked at what threats a mobile agent imposes on a hosts. As we summarize that each host in a system must utilize an efficient control protocol. There are numerous existent protocols developed to achieve one or more components of an efficient access control procedure. With each method presented below, we also demonstrate what its main achieved purpose is. Before proceeding further, a description of each component of an efficient control protocol is described as follows:

- **An authentication scheme:** The main purpose of an authentication scheme is to prove a mobile agent ownership or a mobile agent's sender (i.e. the host who dispatched the mobile agent) by utilizing cryptography functions (e.g. digital signature, hash function, message authentication code).
- **A mobile agent code's investigation:** According to [Gelbart04], code investigation is a method to verify a mobile agent's code according to safety regulations created by the host. To be more explicit, code verification is to ensure that the code does what it is supposed to do and nothing more.
- **Resource provision:** This can be described as a resource access allowance. To be more specific, it is a method by which a host grants each mobile agent an access to its resources according to the degree of authorization containing in a credential (for example, a credential of a mobile agent can be considered as its owner signature on its code. Other types of credential can also be applied) carried by each mobile agent. For example, a host restricts access to its file database only mobile agents coming from a particular group of hosts.
- **A mobile agent's activity monitoring:** After having received permission to access some particular resources, a mobile agent can pursue its tasks on the host. The host must monitor the mobile agent's activity so that it does not endanger any part of the host system (i.e. host itself or other agents).

Signed Code

Code signing is a fundamental technique of confirming the authenticity of objects (e.g. mobile agents) and its integrity, using digital signatures. The authenticity of a mobile agent can be proved when checking the digital signature on a particular mobile agent's code. The examiner will explicitly know (we assume that the examiner knows the signer's public key) either the creator's identity (i.e. owner of the mobile agent) or some entities' identities (i.e. hosts that have reviewed or dispatched the mobile agent). Its integrity serves as an assurance that the code has not been altered. Code signing involves public key cryptography. To make code signing methods be applicable, a public key infrastructure must be available. From [Microsoft03], we learn that this method cannot ensure that the code is safe to execute. For example [Janzen00b], introduces Microsoft's Authenticode, a common form of code signing, that enables Java applets or Active X controls to be signed for ensuring users that this software has not been tampered with or modified and that the identity of the author is identified. We conclude that the main purpose of this method is to serve as an authentication scheme, however it does not guarantee that the consequence of the execution of the mobile agent code is harmless or not harassing.

Path Histories

This method was introduced by [Chess95]. The idea of path history is to maintain an authenticable record of previously visited hosts during a mobile agent's lifetime. A host uses the path history to decide whether it should grant its services under the restraint of where a mobile agent has been migrating before it arrives at its site. To build a path history, it requires each host in a mobile agent's itinerary to append a signed entry to the path indicating its identity and the identity of the next host to be visited and to provide the complete path history to the next host. Obviously, the main drawbacks are, that the cost of path verification becomes higher as the path history increases and that some malicious hosts in the itinerary might modify or not append their signed entries to the received path history. We conclude that the main purpose of this method is to provide a combination of an authentication scheme and resource provision.

Proof-Carrying Code (PCC)

This approach was introduced by [Necula98]. It requires the code producer (i.e. the mobile agent's owner) to officially provide a formal proof that the code adheres to a security policy defined by the code consumer (i.e. the host). Upon receiving the code and its formal proof sent by the code producer, the code consumer validates the code. The result of validation leads to the claim that the code is safe to execute or not. The PCC ensures the safety of code execution. To prove the code, no cryptographic techniques or external assistance is required. In addition, the PCC is secure against code or proof's tampering. In other words, any tampering on either the code or the formal proof results in either a verification error or, if the verification succeeds, a safe code transformation. Before becoming widely used, there are two main limitations found in this approach namely proof generation and time consumption of proof validation process, which must be resolved. We conclude that the main purpose of this method is to provide the assurance, that the code is safe to execute.

Software-Based Fault Isolation

The sandbox model [FM96] aims at limiting a mobile code in such a way, that it cannot damage the local execution environment. To achieve this, the interpreter defines the security policies for the local as well as the remotely originated code. Whereas local code is usually granted access to all resources (I/O, File system, etc.) the remotely originated code runs in the sandbox and is thus subject to higher restrictions with respect to resource access. It may, for instance, not be allowed to access system commands. This approach limits the useful functionality remote code can implement for example to display graphics and thus is not attractive for real-world applications.

State appraisal

The idea of this approach (see [Farmer96]) is that a host derives a mobile agent's degree of authorization from its state information using a state appraisal function created by the agent's owner. A state appraisal function computes a set of privileges to request as a function of the mobile agent's state. After the state appraisal has determined which permissions to request, a host uses a resource provision method to determine which of the requested permissions will be granted. This approach is developed to cope with illegitimate mobile agent's state information alterations. For example, while a mobile agent is migrating among hosts, it brings code, static data, collected data and state information. State information of a mobile agent refers to the dynamic data created during the execution of a mobile agent and used as an input to the computations conducted on the next host. After receiving a mobile agent, a host checks the mobile agent´s state information to be sure that the state information was not tampered with before granting an access to its resource. This method serves as an authentication scheme for resource provision.

10.2.4 A Survey of Mobile Agent Protection Methods

The approaches to protecting a mobile agent can be broadly classified into two main mechanisms [Karnik98]. Detection mechanisms attempt to detect attacks on a mobile agent. Prevention mechanisms try to make it impossible (or at least very costly) to attack a mobile agent. Protection methods can be categorized into two types, according to their purpose as follows:

- **Type 1:**Aims to reduce the chance of migrating to a malicious host or restricts migrating to only trusted hosts. Normally, a protection method of this type utilizes the notion of trust. The notion of trust has been long recognized as being of paramount important for secure systems. We call a method of this type a trust-based computing method. Some methods are discussed below.

- **Type 2:**This can be classified into two aspects, namely a protecting mobile agent's components and protecting the mobile agent's execution. The first aspect refers to the protection of data, variables and code. The second aspect can be categorized into two issues namely; handling of infrastructure failures and ensuring the correctness of a mobile agent's execution. In [Pleish04], Pleish et. al. explain infrastructure failures as follows. Infrastructure failures imply the premature halting of processes (i.e. hosts, communication links). For example, a failing host causes all agents running on it to fail and a link failure causes the loss of all messages or mobile agents currently in transmission on this link. Without infrastructure failures, the correctness of a mobile agent's execution implies that a visited host must faithfully execute a mobile agent according to the mobile agent's code. Generally, most of the approaches in type 2 are strongly designed by using cryptography techniques (i.e. encryption/decryption algorithms, private and public keys, digital signatures, digital time stamps, and hash functions). Some methods are presented below.

Trust-based Computing Methods

A method, which falls into this category, involves either creating a trusted environment or equipping a mobile agent system with a trust model. A trusted environment is a domain in which a mobile agent can roam without being threatened by any entities in the same domain. A trusted environment can be implemented by using a security policy, tamper resistant hardware or external trusted hosts. A trust model can be explained as an efficient method for assessing each interaction partner's trustworthiness (e.g. trust value).

- **Tamper-resistant hardware [prevention], [Wilhelm99]:** The idea is to execute a mobile agent in a tamper-resistant hardware added to conventional computing systems (i.e. untrusted hosts), which can defend potential physical attacks. One major drawback is that a mobile agent can migrate only in a closed environment where every host must be equipped with tamper-resistant hardware.

- **Trust model [prevention]:** Today this is one of the most popular research topics in the area of e-Commerce and Peer to Peer applications. The goal of a trust model is to calculate a trust value of the entity to be used as a quantity of measurement in a decision making process. In other word, we want to assess how trustworthy in a certain aspect the entity is, given some information (e.g. the entity's reputation, direct experiences with the entity, ... etc.). After calculating an entity's trust value, we use it as one of the components (in eBay, such a quantity might be the price or the delivery time, and in a mobile agent system, it might be the service quality) to decide whether we will conduct an interaction (in case of e-commerce, an interaction means trading and in case of mobile agent systems, an interaction means migration to the host) with the entity or not. In [Derbas04], Derbas et.al. present TRUMMAR a trust model for mobile agent systems based on reputation. In TRUMMAR, a host A which wants to send a mobile agent C to another host B in order for the agent to accomplish its task does this, only if host B is trustworthy in providing a computational environment which will not attack (i.e. delete, corrupt, manipulate or misinterpret the agent's code, data or variable) the mobile agent C. In order to assess B's trustworthiness, host A combines host B's reputation from many resources (e.g. its neighbors, friends) with B's reputation from its own perception derived from its direct experiences with B, to came to a summarized reputation value of host B. Host A will compare this summarized reputation value with the predefined threshold value to decide whether host B is trustworthy or not.

Cryptographic-based Methods

For this category, there are numerous methods that have been proposed so far. First we present some methods aiming to protect mobile agent's components as follows:

- **Data protection methods:** Many methods, for example, sliding encryption [Young97], partial result authentication code (PRAC) [Yee99], Chained signature [Karjoth98], have been developed to ensure a mobile agent's data integrity.

Data integrity, in this context, means the integrity of partial accumulated results performed by a mobile agent on each of the visited hosts. Normally a data protection method is utilized in a shopping scenario, i.e. in which a mobile agent roams in a fixed/undetermined itinerary of vendor platforms in search of some specific item on behalf of its owner. The offers provided by each visited vendor can be viewed as a partial accumulated result. Without protecting partial accumulated results, a malicious vendor can perform many types of attack.

– Sliding encryption [prevention]: In [Young97], Young et. al. propose a new mode of RSA public key encryption (the sliding encryption) to ensure the integrity of collected data under a limited storage situation.

– Partial Result Authentication Code [detection]: Yee [Yee99] proposes the Partial Result Authentication Code (PRAC) to ensure the integrity of the collected result. This method provides the agent with a set of secret keys used to calculate a MAC (Message Authentication Code) on the result of each host, using a one-way function to produce the key associated with the current host from an initial secret key given by the originator. PRAC comprises of the result and its MAC. The agent erases the secret key associated with the current host before migration. Yee defines forward integrity, which implies that the first visited malicious host cannot modify or forge any PRACs of already visited hosts.

– Chained Signature [detection]: In [Karjoth98], Karjoth et al. publish a family of protocols that aims to protect the integrity and confidentiality of data collected by a free roaming agent. In their paper, they extended Yee’s protocols and defined a set of security properties that has a higher degree of security. They extend public verifiable forward integrity from [Yee99] in the following manner “Anyone can verify the offer o_i by checking whether the chain is valid at encapsulated offer O_i”. In other words, every visited host is able to verify the integrity of the encapsulated offer by using a one time public/private key pair generated by the previous host. This property is very useful since not only the originator but also the visited host on the agent’s itinerary can detect the tampering.

– Route Protection [prevention]: Assume that each mobile agent visits a set of hosts according to a predefined itinerary given by its owner. A predefined itinerary can be viewed as a restriction for each mobile agent to visit only trusted hosts. Thus it becomes necessary to protect the given route. In [Westhoff99], nested encryptions and signatures are used to provide increased security to the agent´s route. The basic idea is to sign and encrypt host addresses iteratively to attain a high level of security. A flexible itinerary, which allows dynamic adaptation and expansion during the execution of the agent, is preferable. A shortcoming of this flexible itinerary is that it is vulnerable to corruption of hosts specified in the itinerary; a corrupted host can modify the itinerary or attack other hosts in the itinerary to cause denial of service to the agent. There are some proposals [Westhoff00], [Ferrer01] which attempt to prevent this vulnerability.

- Environmental key generation [prevention]: This method protects the integrity of a mobile agent´s code. In [Schneier98], a mobile agent carries an encrypted code. In order to obtain the particular decrypting key, the host environment must contain a predefined condition. The idea is that constructing agents in such a way that upon encountering an environmental condition (i.e. via a matched search string), a key, which is used to decrypt some encrypted executable code, is generated. The environmental condition is concealed through a one-way hash or public key encryption of the environment trigger.
- Computing with an encrypted function [prevention]: This is proposed in [Sander98]. It offers a similar protection, as can be achieved by using a tamper resistant hardware. The idea is to encrypt a function f to obtain some other function $E(f)$, which hides the content of function f. This encrypted function $E(f)$ can then be implemented as a program $P(E(f))$, which is interpreted as a mobile agent and sent to the host. The host then executes $P(E(f))$ on an input x which leads to the encrypted result $P(E(f))(x)$. Since the host does not know what function it actually calculated, it cannot meaningfully tamper with the code or its execution and is restricted to random modifications. The encrypted result is returned to the owner who can decrypt and obtain the desired result of the function f (i.e. $E^{-1}\ P(E(f))(x) = f(x)$). The idea is restricted solely to polynomials and rational functions. In addition it does not allow continuous interactions between an agent and a place (at the host). This approach is thus not well suited for general use.

In [Collberg02], Colleberg et. al. identify three methods that are adopted as alternative defences against attacks on intellectual property contained in software namely; Code obfuscation, Watermarking, and Tamper-Proofing. Undoubtedly, all of them can also be adapted to protect mobile agents.

- Code Obfuscation [prevention]: This is a process that renders software unintelligible but still functional. In other words, it is a method of transforming an original agent's code into another code, which is functionally identical but very hard to read and understand. It ensures security through obscurity. In [Hohl98], Hohl proposes a blackbox security using code obfuscation. Since obfuscation cannot accommodate perfect protection, a time limited variation of blackbox is introduced. After the interval elapses, the agent becomes invalid. This method is suitable for an application, which does not convey information. A main drawback is the lack of methods to calculate a reasonable time period for the mobile agent execution.
- Watermarking and Fingerprinting [detection]: Watermarking is a process, which embeds a copyright notice in the software code to allow the owner of the software to assert his intellectual property rights. The fingerprinting is a

similar technique that embeds a unique customer identification number into each distributed copy of an application in order to facilitate the tracking and prosecution of copyright violators. In [Esparza03], Esparza et. al. propose mobile agent watermarking and mobile agent fingerprinting which first aims at the assurance of the integrity of the execution and not at property right protection but later not only provides detection of attacks performed during the mobile agent's execution but also detects collusion attacks launched by a group of hosts. Mobile agent water marking can for example be achieved as follows; a mark is transferred to the agent's results during the execution. However, the mark is disguised as part of the real results to the executing hosts. After migrating back to the owner, each result is verified. When the marks are differ from the owner's expectation, this means that the host has modified the agent. In mobile agent fingerprinting, the embedded marks are different for each host. During the verification process of each result, if an altered mark is detected, the host generating this mark is malicious. The authors claim that by assigning different marks to each host makes the possibility of collusion harder. In both methods, after a malicious host is identified, an external authority must punish it. Drawbacks of both methods are increase in code and data size and the need of an external entity.

  – Tamper-proofing method [prevention]: This causes a program to malfunction when it detects that it has been modified. Thus it may be self-protected.

- **Execution Protection and Infrastructure failure handling:** There are some approaches in protecting the mobile agent's execution, namely a host is obliged to generate a log of the operation performed by the mobile agent while residing there (e.g. execution tracing)

  – Execution tracing [detection]: [Vigna98], introduces such a method for detecting unauthorized modifications of an agent through the faithful recoding of the agent's behavior during its execution on each visited host. This technique requires each visited host involved to create and retain a non-repudiable log or trace of the operations performed by the agent residing there, and to submit a hash of trace upon conclusion as a trace summary. A trace comprises of a sequence of identifiers corresponding to the executed statements in the agents code and affixed with the digital signature of the executing host. A secure protocol is also developed to convey agents and associated security related information among the various parties involved, which may include a trusted third party to retain the sequence of trace summaries for the agent's entire itinerary. If a suspicious result occurs, the appropriated traces and trace summaries can be obtained and verified, a malicious host thus is identified. This method can only detect a manipulation after a mobile agent returns to its owner. Only when the owner becomes suspicious of a result, then the verification process is activated. Execution tracing has a number of drawbacks, e.g. the size and number of logs to be retained, the verification process is triggered sporadically (i.e. based on the owner’s suspicion). As further reading on extended execution tracing we recommend[Tan02], [Tan04].

– Server and Mobile agent Replication [detection]: Infrastructure failures can only be minimized by introducing redundancy. In [Minsky96], a set of independent replica hosts is used to provide parallel computation. After the execution, the hosts vote on the result. A well-known problem when utilizing server and mobile agent replication is multiple executions of the mobile agent. This problem plays a crucial role in applications e.g. money transactions, shopping. In [Rothermel98], an agent execution is defined to be “exactly once” if the entire sequence of its stages is eventually performed, and all operations of each stage are executed exactly once. Whilst this is not a problem of applications, e.g. reading data. As an example, Rothermel et.al. propose a protocol for a scenario where a user launches a mobile agent to make a flight and hotel reservation for a forthcoming business trip. The agent is expected to make both reservations if possible, and in any case return a status message back to the user. This protocol applies ‘observer nodes’ for monitoring the state of the currently executing node, which take over the execution when this node is not available. A voting procedure integrated in the processing ensures the “exactly-once” semantics.

11 Copyright Protection

Not everything that is on the Internet is public domain and may be used without permission from the creator/owner. For our discourse copyright is a protection that covers published work over the Internet. There are a number of technologies that can be used to protect and enforce copyright. Two of these technologies we have already encountered in the last chapter are watermarking and fingerprinting.

Copyright protection becomes more difficult in the digital world than in the analog world, since a copy of digital data can be easily, inexpensively and quickly marketed and distributed over the Internet. Moreover, each copy is not distinguishable from the original[107].

In general, there are two approaches which enables copyright protection:

1. **Usage control:** In this approach, each usage of the protected material, such as viewing, playing or printing, is controlled by some authorized rendering hardware or software [Opp00]. Pay-TV and video-on-demand are two applications that are protected according to the usage control approach.

Digital marks

2. **Digital copyright labeling techniques:** Digital copyright labeling techniques embed **digital marks** into protected material. These digital marks hold as a distinctive symbol for the copyright related information such as origin, owner, content, or recipient. The idea behind digital copyright labeling techniques is to allow unlimited copying and usage of the protected material, but to provide evidence for copyright violation. In this approach, in turn, two types of digital copyright labeling techniques can be distinguished:

   Ownership labeling

   a. **Ownership labeling:** Ownership labeling is referred to as watermarking. A multimedia document is marked with a label that uniquely identifies the copyright holder. A misuse of the protected document can be detected by the determination of the legitimate owner of the protected material and the corresponding copyright.

   Recipient labeling

   b. **Recipient labeling:** Recipient labeling is referred to as fingerprint. A multimedia document is marked in a manner that allows its distribution to be uniquely traced [Opp00]. In other words, an illegitimately redistributed copy of the protected material can be traced back to its original recipient.

11.1 Watermarking

Watermarking

Digital **watermarking** is a technology for embedding various types of information (digital marks) in digital content. It is an adaptation of the commonly used and

107 In the analog world, as the number of copies increases the quality of these copies decreases. On the other hand digital original and copies are not distinguishable!

well known paper watermarks. For example, paper bank notes or stock certificates contain watermarks. Otherwise, they could be easily copied and (mis)used. As a consequence thereof, the trust in the authenticity of these paper bank notes or stock certificates would greatly be reduced, resulting in a big loss.

More precisely, digital watermarking is a digital signal added to digital data, such as a multimedia document, that can be detected or extracted later to make an assertion about the data. It can serve several purposes, such as ownership assertion, authentication and integrity verification, content labeling, usage control, and control protection [Opp00].

A watermark can be categorized to be visible or invisible[108]:

Visible watermark

1. A **visible watermark** is – as the words say – visible to the eye. Example of this category is a company logo (see Fig. 11.1-1).

Invisible watermark

2. An **invisible watermark** is imperceptible by the user but can be read by a computer with the proper decoding software. This is analog to seals or marks on paper currencies that are only visible when held up against light.

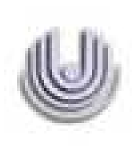


Fig. 11.1-1: An example of a visible watermark: The logo of the University of Hagen.

Furthermore, watermarks can be robust or fragile, depending on the application.

Fragile watermarks

1. **Fragility** means that the watermark does not survive tampering by any image-processing transformation. A fragile watermarking is convenient to image integrity checks where each modification on the image content must be detectable.

Robust watermarks

2. **Robustness** refers to the capability of the watermark to survive any manipulations of the media, such as lossy compression, scaling, filtering, and requantisization, etc. A robust watermarking is required for ownership assertion.

Digital watermarks can be classified in public or private watermarks, too:

108 All our next discussion will be referred to watermarking for images, but is valid to watermarking for other multimedia documents such as audio or video. In this regard, we will speak about audible and inaudible watermark in the case of audio material.

Public watermark

1. A **public watermark** can be detected and read by anyone using an appropriate detector software without the need of any secret information. For example, a detection of copyright violation in images published on the Web can be accomplished with public watermarking techniques. Therefore, we can use mobile agents (detector software) to perform identity checks for images at locations of the Web.

Private watermark

Secret key watermarking

Public key watermarking

Insertion

Extraction

Detection

2. A **private watermark**, however, cannot be detected and read without accessing some secret information. Of course, this is also accomplished by using an appropriate detector software. Private watermarking provides a higher level of security but is less practicable in comparison to a public watermarking. It is clear that the secret information must be communicated and distributed to a user or third party in a secure manner, in case of the watermark detection is not carried out by the image owner. So the private watermarking scheme is also suitable to demonstrate ownership of content in case of misuse. Further, private watermarking can be subdivided into **secret key watermarking** and **public key watermarking** according to the keys used for **insertion**, and **extraction** or **detection** (see Fig. 11.1-2).
   a. Secret key watermarking: The same (one) key is used for watermark insertion and extraction or detection. This secret key must also transported via a secure channel.
   b. Public key watermarking: A private key of the image owner is used for watermark insertion, whereas its public key is used for watermark extraction or detection.

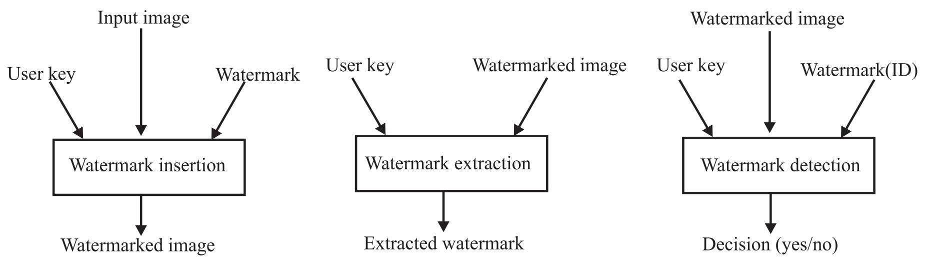


Fig. 11.1-2: The watermarking process including watermark insertion, extraction and detection [Opp00].

Watermarking techniques must be carefully analyzed and developed in order to prevent all possible attacks against them. In general, four classes of attacks are possible:

Robustness attack

1. **Robustness attacks:** Robustness attacks aim to diminish or remove the presence of a mark in a watermarked image without harming the image [Opp00]. Such attacks can be made by means of operations such as data compression, filtering, resizing, etc.

Presentation attack

2. **Presentation attacks:** In presentation attacks, the mark is not removed but the image is manipulated in such a manner that the mark cannot be detectable anymore.

3. **Interpretation attacks:** In interpretation attacks, the watermark is so manipulated that it will have an invalid or multiple interpretation from watermark evidence point of view. If this occurs, the watermark ownership cannot be provided anymore.

Interpretation attack

4. **Legal attacks:** In legal attacks, an attacker does not use technical means against watermarking techniques. He just uses judicial means[109] to despair evidence of ownership presented.

Legal attack

11.2 Fingerprinting

Fingerprinting enables to make each sold or distributed copy of the protected material unique. This uniqueness can be guaranteed through the introduction of individual marks in each copy. In this manner, the provider of the protected material can detect from the fingerprint, the illegally redistributed copies.

Essentially, the techniques and attacks which are introduced for watermarking are the same as for fingerprinting. Most fingerprinting techniques are symmetric, i.e. the provider of the protected material and the buyer know the fingerprinted copy. Consequently, if another copy with the same fingerprint is made, responsibility of illegal copy cannot be assigned to one of them. In other words, **non repudiation** cannot be provided by symmetric fingerprinting techniques. To solve this problem, asymmetric fingerprinting schemes have been proposed. They enable only the buyer to know the document with the fingerprint. However, if the provider discovers a copy of the protected document, she/he can identify its buyer, and prove that such buyer is responsible for the illegal copy. For example, in the technique used in pay-TV, a video stream is transmitted over a broadcast channel in encrypted form, (asymmetric) traitor tracing schemes as application of asymmetric fingerprinting enable to trace people who abuse the broadcast encryption scheme by allowing illegitimate users to decrypt the data stream.

Non repudiation

109 For example, through different interpretations of copyright laws.

12 Intrusion Detection

12.1 Introduction

Physical intrusion

Basically there are three possibilities for an intruder to get into somebody elses computer system. Ones can get in physically (**physical intrusion**), provoke a Denial of Service (DoS) by disconnecting important network elements such as routers, servers, discs, etc. or install a sniffer to record the whole network traffic looking for passwords.

System intrusion

A user who already has an account on a system may be interested in extending his privileges. In most cases users with low level privileges try to extend their privileges (**system intrusion**) and in the ideal case want to become root (Unix) or gain administrator privileges under Windows.

Remote intrusion

Footprinting

Remotely attacking a system can be done in diverse ways. One possibility is to exploit software bugs in running services at the target system and generate a buffer overflow (see Section 12.2.2.2) to gain access to the system. Another way is to make running services unavailable by, for example, flooding the target system with requests which are not processed (Denial of Service). In this course unit we will only deal with system and remote intrusions, but before attacking a system, a clever intruder will first collect as much information as possible about the target system. How this can be done is shown using a brief scenario (**Footprinting**).

Port scanning

If an attacker wants to access a system, the first thing he/she has to do, is to determine the topology or structure of the system, by checking which computers are alive. To do this, he/she generates an Echo request, generally known as PING (Packet Internet Groper). Let us assume that the target host has the IP address `192.168.1.2`. The opponent then generates a `ping 192.168.1.2`. If the target host is alive then he/she will generate an Echo reply message. The attacker now knows that the host is alive. The next step is to determine which services are offered by the target host by generating a so-called **port scanning**. The result of a port scanning is the number of each port where a service is running. Now the attacker identifies those services which are reported to have bugs. Many known vulnerabilities in software are published in the internet (see CERT http://www.cert.org and Micosoft Security Bulletin Search http://www.microsoft.com/technet/security/current.aspx). Very often, when a bug has been found in software, patches are published in order to solve the problem and close the hole. But there is only a small amount of administrators that install patches. According to a recent study carried out by the gartner group (www.gartner.com) 90 % of all attacks can be avoided if all patch-levels are kept up-to-date and another 90 % of future cyberattacks will exploit known security flaws, for which a patch is available or a solution known.

Fingerprinting

After identifying active hosts and running services, since many security holes are platform dependent or related to a specific operating system, the next step in the attacker roadmap is to determine the running operating system, commonly known as **Fingerprinting**.

The implementation of the IP protocol stack depends on the type of operating system, especially when it comes to a reaction to crafted packets e.g. a packet where the SYN and FIN flag have both been set. If a database of the reactions from each operating system is set up, crafted packets can be construed, sent to the server and based on the reaction of the server, it is possible to determine which operating system is in use. The automated scanning tool Network Mapper (NMAP) (www.insecure.org/nmap/) provides options to find out what the underlying operating system is.

Currently, diverse tools are available for the automatic scanning of vulnerabilities in networks. Nessus (www.nessus.org/) and the Security Administrator Tool for Analyzing Networks (SATAN www.porcupine.org/satan/) are some of these tools. The tools can be of great importance for the network administration in looking for security threats, but they can be used by intruders as well to randomly scan the internet looking for exploitable vulnerabilities. However, after the attacker has sufficiently explored the system, brought together all the information needed and eventually found security holes in the system, he/she initiates the next step by trying to exploit the known vulnerabilities and start an attack. But before dealing with these attacks, it will be interesting to investigate what enables them.

12.2 Intrusions

Basically there are a multiplicity of reasons which allow an intrusion to happen. The following are worth mentioning:

- Configuration errors
- System or protocol related errors
- Programming errors
- Carelessness towards attackers.

Configuration errors

A bad configuration of a component can have a great impact on the whole system or network. If we take a web server as an example a misconfiguration of the configuration file *httpd.conf* could allow clients to access files they are normally not allowed to. Many software applications are configured with default passwords. In some cases these passwords are not changed and since they are known to attackers, they can be used to gain access into the system.

Protocol related errors

Protocols related errors result from the fact that by the design of the IP protocol stack version 4 (IPv4), security related aspects had to take a back seat. However the situation is different now. The internet has become a tool of great importance for private and business purposes and therefore attractive to intruders. The implementation of IPv6 (see Kurseinheit 13, Kommunikationsnetze- und Protokolle) was supposed to solve some of the security related problems. However the proliferation of IPv6 has not been that good yet mainly because of the complexity related to it‘s deployment. In this course unit, protocol related attacks are explained under the assumption that IPv4 is in use [RFC 791].

Programming errors

Programming errors are often buffer overflows. A buffer overflow is a non definable system state which can be used by attackers to get into the system by for example crashing higher privileged programs (see Section 12.2.2.2).

Social engineering

The last point deals with the careless handling of passwords: whether they are written down and left beside the computer or it is easy for an attacker to guess them using **social engineering**. Social Engineering occurs when someone tries to abuse the trust relationship by for example claiming to be the administrator to gain access to the password. As a result of the above and other reasons; attackers can get into systems and cause serious damage.

Since there are diverse attacks which can append, only some of the types will be dealt with namely protocol related attacks, remote access attacks and malware in more detail.

12.2.1 Protocol related Attacks

These attacks are enabled by week protocols and related security mechanisms. We categorize the attacks according to the seven layers OSI model and try to analyse the reasons enabling them, starting with (Address Resolution Protocol-ARP) related attacks.

12.2.1.1 ARP Attacks

MAC address

The internet protocol was developed, in order to make different network technologies compatible with each other. The IP address used to route packets from sender to receiver is a logical address, which means that it is not bound to a physical medium and is an abstraction of the underlying physical network. This capability enables networks, which are different from each other from a technological point of view to be connected with each other and extended to a global (world-wide) network. Here, a difference has to be made on the one hand between services e.g. e-mail, file transfer, domain name service running above the TCP/IP layers, which use the IP address and the physical transport medium on the other hand. Inevitably the data has to be fed into the physical medium. The (**Medium Access Control**-MAC) address - a world-wide unique address which is recorded in the network interface card - is used for the communication on an Ethernet network. But since higher layer services operate with IP addresses, a resolution mechanism is needed to map the IP address into a physical unique address. This function is performed by the ARP protocol [RFC 826], which runs beyond the IP protocol in the OSI model. To send a message to a receiver, the sender needs his/her physical address. Since the sender does not know the address of the receiver a priori, he looks at his ARP cache (address resolution table) to check whether the address of the receiver is available or not. If not, the sender then generates a broadcast request to all hosts which are connected in the local network.

The broadcast request consists of: the IP address of the sender, the MAC address of the sender and the IP address of the receiver. All hosts in the Local Area Network (LAN) can see the request, those who are not concerned ignore the request and only the host whose MAC address corresponds to the generated IP address responds to the request and sends a reply to the sender per unicast. The relation (association) between MAC and related IP addresses is stored in the ARP cache so that multiple requests and the related waste of bandwidth can be avoided. Since at the ARP level, no authentication mechanisms have been implemented, an attacker can use the absentee authentication mechanism to reply to a request and generate a resolution which is not correct (see Example 12.2-1).

ARP spoofing

Example 12.2-1:
In this example Host A want to establish a connection to receiver D (see Fig. 12.2-1). He generates a request as follows: `Who has` IP_D`? tell` IP_A (1). On receiving the request, the opponent C generates a reply by sending it MAC address MAC_C to sender A (2). The sender A has no possibility of verifying whether the generated MAC address MAC_C in fact corresponds to the IP address of D. Attacker C has sucessfully spoofed the address of Host D. Host A and attacker C update their ARP tables (3).

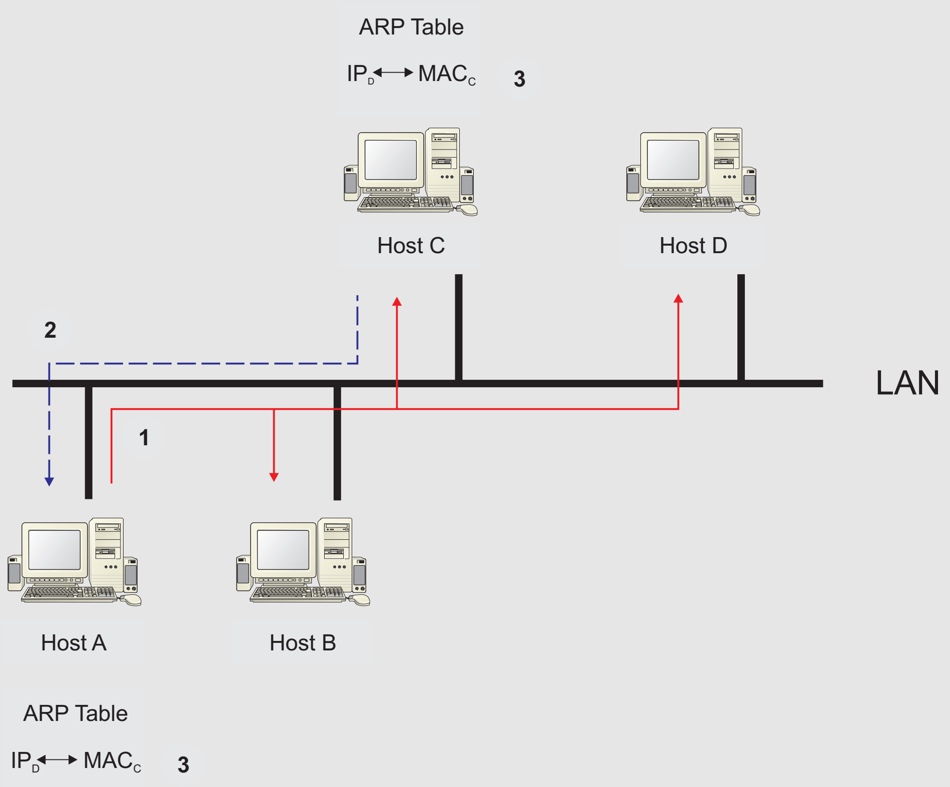


Fig. 12.2-1: ARP spoofing

This attack works best if the host whose IP address has been spoofed, is disabled. Since the records in the cache are cleansed cyclically, the attacker will

ARP cache poisoning

have to update the cache records (**ARP cache poisoning**) if the attack is to be efficient. This attack can cause enormous damage in the local network, e.g. a man-in-the middle attack using a notebook in the LAN[110]. ARP spoofing works even in a switched environment and there are diverse utilities available to perform ARP spoofing. Using the tool Ettercap, a sniffer for switched LANs (see http://ettercap.sourceforge.net/) the cache poisoning process can be automated.

ARP spoofing can be avoided by only allowing static ARP tables. With static ARP tables a change in the MAC-IP pairing is not possible. This solution has a major drawback, it can only work in small networks. In large networks where addresses are assigned dynamically, ARPWatch can be used to keep track or monitor the MAC-IP pairing and inform the administrator when changes occur.

12.2.1.2 IP Spoofing

As can be seen from the structure of an IP datagram version 4 (see Fig. 12.2-2), there is no field for authentication. This means if a host receives a packet from another host, the receiver does not have the possibility to check out whether the IP address inscribed in the field is really the address of the sender host. The receiver just assumes the packet originating address is from the sender, but there is no proof that it really comes from him. In other words, there is no mechanism to verify the authenticity of the sender IP address inscribed in the datagram, with the disastrous result that an attacker can either misuse IP based authentication or cloud his identity by using an address which does not belong to him. This phenomenon is illustrated in Fig. 12.2-3.

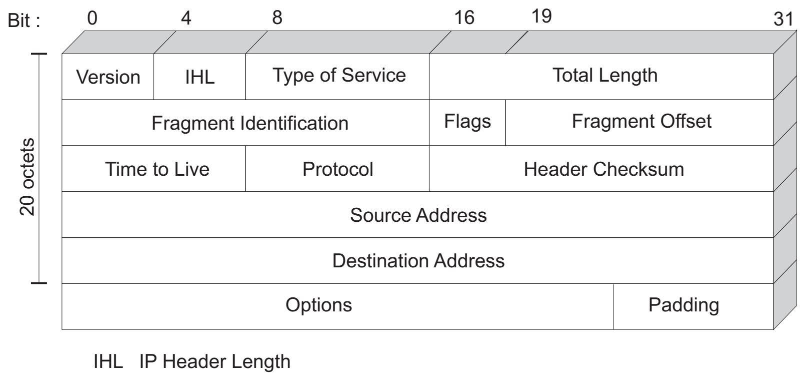


Fig. 12.2-2: IP datagram

110 The attacker connects himself to the path between the client and the server. For the attack to be successful he has to convince the client that he owns the server MAC address and convince the server that he owns the client MAC address. In doing so, all packets from the client to the server and from the server to the client will be redirected to the attacker.

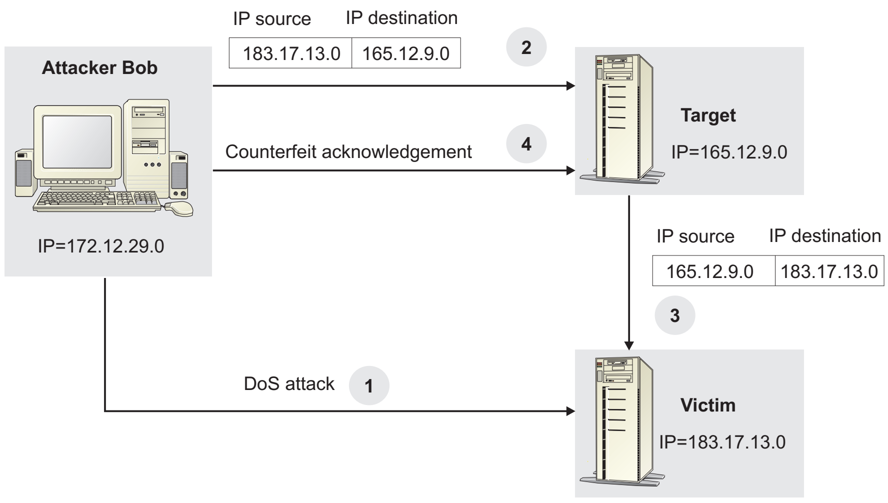


Fig. 12.2-3: IP spoofing

IP spoofing

We have three participants: an attacker (Bob), a trusted host (Victim) and a server (Target). First of all Bob disables the victim by issuing a DoS to the trusted host, so that it can not respond to any request otherwise it might see packets from the target and will send an RST packet to tear down the connection. Bob then generates a packet with the IP address of the trusted host `183.17.13.0`, pretending to be the trusted host. Since the victim is trusted by the server the server checks the IP address and believes it really comes from the victim. But in reality it comes from the attacker. The packet from the server is sent to the victim, but since the victim has been taken off, it can not respond. The attacker does not see the packet from the server, but he can further process and perform some operations on the server.

Trust relationship

IP based authentication can be bypassed through the misuse of **trust relationship**. This trust relationship was implemented in the Berkeley Unix operating system for client/server protocols belonging to the so-called *r family* (e.g. rlogin, rcp, rsh) to alleviate jobs for users and administrators. The trust relationship can be carried out either host wide, using the file /etc/host.equiv, which contains the name of the hosts considered to be trusted or user wide, in which case the file .rhosts is used and trusted users will not be prompted for passwords. IP spoofing is a blind attack meaning that the attacker generally does not receive the response of the server. In order to receive server´s generated packets the attacker must be able to access a router along the path between him and a server or carry out a **source routing attack**.[111]

Source routing attack

IP spoofing can be averted by using IPv6 [RFC 2460] where authentication fields have been implemented. IP spoofing is a dangerous attack, which is often used as a starting point in order to perform further attacks for example DoS attacks or TCP connection hijacking. The latest will be dealt with in the following section.

111 The attacker defines the path along which the packets are sent to him. Currently most routers do not allow Source Routing.

12.2.1.3 TCP Connection Hijacking

Hijacking a connection means stealing a connection which has already been established. Since all security related mechanisms such as password control happen before the connection is set up, hijacking is very powerful. To understand TCP connection hijacking, some basic terminology of TCP is needed and will be introduced next. First of all, let us take a closer look at a TCP datagram (see Fig. 12.2-4).

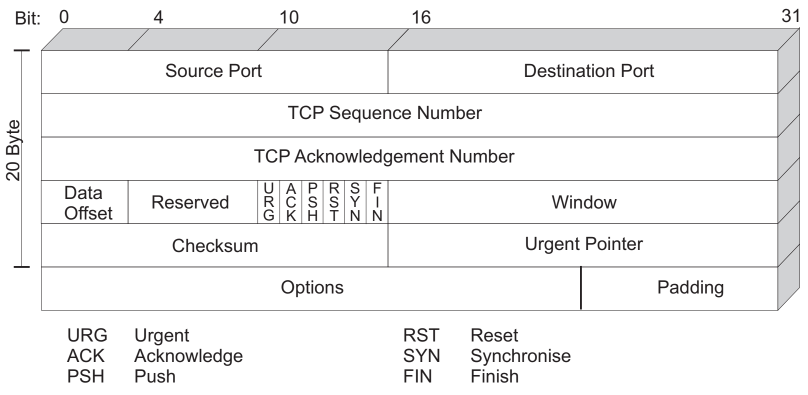


Fig. 12.2-4: TCP datagram

Three way handshake mechanism

The fields TCP sequence number, TCP acknowledgement number and the flags (URG, ACK, PSH, RST, SYN, FIN) are primarily relevant for the understanding of the attack. Generally a TCP connection consists of three phases, the so-called **three way handshake mechanism** (see Fig. 12.2-5). At the beginning the sender generates a synchronisation packet with the flag bit `SYN` set to 1 and the sequence number equals the initial sequence number = X. The receiver confirms the receipt of the synchronisation packet by generating a reply packet -with flags `SYN` and `ACK` each set to 1, Initial Sequence Number Y and $ACK = X + 1$- to acknowledge the receipt issued by the client and to further communicate to the client it's own initial sequence number Y, which will enable a bidirectional communication. On receipt of the packet provided by the server, the client acknowledges it by generating a packet with a flag `ACK` set to 1 and $SEQ = X + 1$, $ACK = Y + 1$. The connection has now been established and the data can be transmitted in both directions. The connection can be either closed by the client or by the server by generating a `FIN` or a `RST` packet.

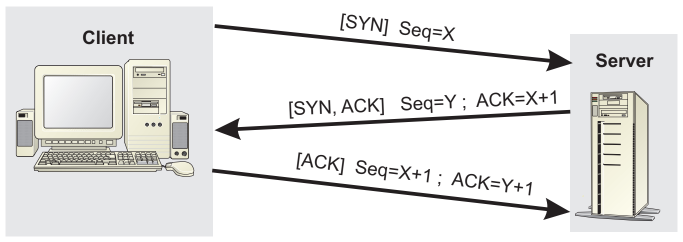


Fig. 12.2-5: Three way handshake mechanism

A closer look at the three way handshake mechanism reveals that hijacking a connection turns to determination of the sequence number of the server[112]. There are many possibilities to gather the sequence number of the server. Firstly if we can access the local network or a router along the path to the server, performing a man-in-the-middle attack, all packets between server and client can be sniffed and the sequence number as well. The second and probably more elegant possibility is to predict the TCP sequence number. The sequence number prediction works because of implementation flaws in relation to sequence number generators which allow attackers to predict TCP numbers using primitive statistical approaches [Zav01].[113] In Fig. 12.2-6 it is shown how the attack works. We have three actors: the opponent (Attacker), the client (Victim) and the server (Target). At first the attacker waits until a connection has been established between client and server (Step 1 to 3). Then the attacker issues a DoS to take off the client (4). Now the attacker has to predict the next sequence number which will be used by the server. To do this the attacker initiates a connection to a harmless port (5) by generating a SYN packet. The server acknowledges the receipt of the packet (6), the attacker has the current sequence number and generates a RST (7). The attacker, using the current sequence number, can now predict the next sequence number as described in [Zav01]. The attacker then initiates a new connection to the server by generating a SYN packet spoofed with the address of the victim (8). The reply of the server is sent to the client (9), it can not respond since it has been disconnected using a DoS attack. The opponent can not see the answer of the server, but acknowledges it with the predicted sequence number (10).

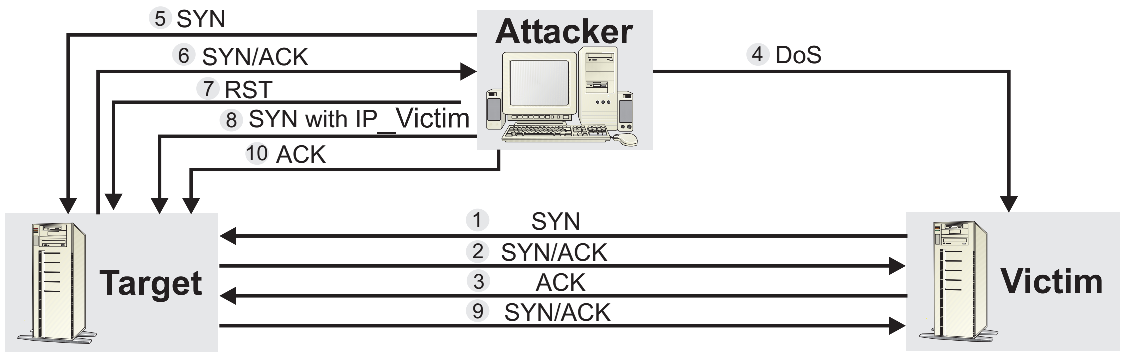


Fig. 12.2-6: TCP connection hijacking

TCP connection hijacking can be prevented if secure sequence number generators are implemented (see [RFC1750] and [RFC1948]).

112 Second step in the three way handshake.

113 For further discussion about random processes especially with regard to security see (RFC 1750 and RFC 1948).

12.2.2 Remote Access Attacks

There are diverse options to remotely access a system, some of which will be discussed in this course unit, namely by gaining a legitimate user password or by carrying out a buffer overflow. Firstly let us take a closer look at password attacks.

12.2.2.1 Password Attacks

In most computer systems, authentication is still carried out using a user name in combination with a password. This basically means that if an attacker is able to come into possession of the password he/she can access the system. There are diverse strategies to gain possession of passwords, some of which will be discussed in the following

- Simple password guessing
- Password sniffing
- Password cracking
- Social engineering.

Password guessing

Simple password guessing is systematically trying to guess the password by entering passwords which are frequently used. Diverse lists with passwords often used are available in the internet. The attacker uses these lists to insert typical login combinations. There is no need to mention that this method can only work in a few cases because most systems will refuse further access following a certain number of access failures.

Password sniffing

Sniffers are network utilities used by network administrators to record all packets which traverse a network. Since there are many applications where the password is transmitted non encrypted e.g. Telnet (Teletype Network) or FTP (File Transfer Protocol), the attacker just reads the whole network traffic and gathers passwords.

Password cracking

Password cracking uses more sophisticated methods, to find out the password e.g. brute force, systematical passwords search or dictionary attacks. In the following a closer look will be taken at brute force and dictionary attacks on passwords.

Bruce Force Attack

Brute force attack

A **brute force attack** consists of trying all possibilities available to find the password. Under the assumption that the attacker has infinite resources and time, each password can be found using the brute force method. In reality, the attacker only has a certain amount of time and resources. Therefore brute forcing a system is not very efficient. Before dealing with the brute force method in detail, let us first take a closer look at how password authentication works.
At the beginning a user is prompted by a login process to provide a combination of user name and password. The login process then compares the provided information with the information stored in the password file. If there is a match, the user is allowed to access the system otherwise the login request is rejected.

If X is the number of printable characters on a keyboard and α the number of maximum characters a password can take, then the password search space is of length: $X^{\alpha} + X^{\alpha-1} + X^{\alpha-2} + \ldots + X^{1}$. If we know the number of operations a processor can process in a given amount of time (MIPS) and we know how many instructions are needed to perform one password search, then we can derive the maximum searching time.

Example 12.2-2:
In many operating systems the password is 8 bits in length. Since the number of printable characters in most keyboards is 95, the password search space can easily be derived and equals $95^8 = 6,63 \times 10^{15} = 6634204312890625$. If we further assume that the opponent has a modern standard PC with an Intel Pentium processor of the fourth generation (Pentium 4) which can perform approximately $8.000.000$ guesses in a second, then it will take $26,296$ years to brute force the system.

Unfortunately searching for passwords is easier than presented in the example above. In reality things look completely different. When a user is given an account on a system, he/she chooses her own password. To avoid having to write the password down on a piece of paper, users tend to choose easy to remember passwords such as names of children, name of pet, etc. Generally user passwords are not chosen according to security principles or password guidelines [Yan01][Yan04] with the predictable result that they are easy to guess.

Example 12.2-3:
The number of printable characters in most keyboards is 95. These characters can be grouped in four dimensions Numeric (10), Alphabetic (26), Upper/Lower case (52) and Keyboard Extended (33). The password search space and the password search time on a pentium IV for 8 character passwords chosen from different search spaces has been summarized in Table 12.2-1 for the following cases:

1. All eight characters are alphabetic
2. All eight characters are numeric
3. Six are alphabetic and two are numeric.

Tab. 12.2-1: *Search space and search time*

	Search space	Search time
1. Alphabetic	$26^8 = 208827064576$	$7,25\,h$
2. Numeric	$10^8 = 100.000000$	$12,5\,s$
3. Combination	$^{114}C_8^2 \cdot 10^2 \cdot 26^6 = 864964172800$	$30,033\,h$

The disadvantage of the systematic password search method from the opponent point of view is the fact that many systems would only give you a couple of attempts to access which means that brute forcing or systematically guessing passwords is nearly impossible. This along with the evidence that well chosen passwords are very difficult to guess led to a more pragmatic solution called *dictionary attack* being used, and this will be discussed next.

Dictionary Attack

To carry out a dictionary attack an opponent needs a dictionary, the encryption algorithm, a powerful processor and the password file. The first three things are quite easy to acquire. Currently there are large amounts of electronic dictionaries available with the most common passwords in use. The main obstacle an attacker has to surmount is to get the password file. In the case of Linux, this password file was previously located in `/etc/passwd`. After several attacks, the file was shadowed and stored in `/etc/shadow` with the difference that, now, only the superuser root can access the password file. An abstract of a password file `/etc/passwd` is shown below. At first the user name is stored in clear (`root`), then comes `x` which stands for the encrypted password. After the password comes the user ID, in this case user root has the number `0`, the group ID `0`, and then comes the Geco field (general information about the user), the home folder of the user with the shell that will be invoked.

/etc/passwd

```
root:x:0:0:root:/root:/bin/bash
daemon:x:2:2:Daemon:/sbin:/bin/bash
lp:x:4:7:Printing daemon:/var/spool/lpd:/bin/bash
bob:x:505:100::/home/bob:/bin/bash
james:x:507:100:/home/james:/bin/bash
peter:x:506:100:/home/peter:/bin/bash
edward:x:510:102:/home/eduard:/bin/bash
```

As can be seen from the shadow password file, users names are stored in clear text and the password is encrypted. The superuser root e.g. has the password `rjABCv9A8PLKu`, and the user bob `aoCFDr2E7IJKs`.

`/etc/shadow`

```
root:rjABCv9A8PLKu:12235:0:10000
daemon:*:3902:0:10000
lp:*:8902:0:10000
bob:aoCFDr2E7IJKs:12001:0:55555:3:
james:pdLIDr7F3LGBt:12001:0:55555:3
peter:uiFDSi4H9RTWw:12001:0:55555:3
edward:tzLFKz6R79KPs:12001:0:55555:3
```

114 $C_8^2 \equiv$ possibilities of choosing 2 out of $8 = \frac{8 \cdot 7}{1 \cdot 2}$.

Shadowing the password file has led to an increase of system security, since nobody else except the superuser root is allowed to access the shadow password file but in the past it was reported that there are bugs on running processes which allow attackers to read the shadowed file [Sch01]. However, if the attacker can come into the possession of the password file, he can encipher the dictionary and then compare the ciphered list of common passwords with the shadow password file. If there is a match, then a password has been found. To augment the probability of finding a valid password the attacker uses each word in different variations. This method strongly reduces the password search space as can be asserted from diverse studies which have been conducted in the past in order to increase password security. The work of Robert Morris and Ken Thompson in 1979 is a milestone in testing the power of user passwords [Ken79]. The test was carried out on a PDP-11/70. 3289 passwords were examined and 30 $\%$ of the passwords could be found and the search only lasted 5 minutes.

In another case study of over 14,000 Unix passwords, almost 25% of the passwords were found by searching for words from a carefully formed dictionary consisting of only $3 \cdot 10^6$ words [Mon99].

Password guidelines

There are many approaches to strengthen password security. One is to select passwords randomly. The idea behind this approach is that the random selection of passwords will lead to a better use of the password space. A disadvantage of this approach is that randomly selected passwords are often not pronounceable and thus not easy to remember. Another approach is to use computer generated passwords which are pronounceable. A further method, called proactive password checker suggests checking passwords before taking them into use [Bis95][Blu04]. This means the password chosen is subject to a dictionary attack and if it is broken the user will have to provide another one. Password aging is another approach which consists of forcing the user to change their password after a certain period of time. The benefit of this method is not evident.

12.2.2.2 Buffer Overflows

A closer look shows that this attack leads the current list of the CERT statistics. Below a short list of reported bugs of the CERT statistic:

```
2. June, 2006, Mozilla contains a buffer overflow vulnerability in
   crypto.signText()
20. May, 2006, Microsoft Word buffer Overflow
25 April 2006: Multiple vulnerabilities in DNS implementations
14 March 2006 Microsoft Office routing slip buffer overflow.
```

Before we deal with the attack in greater detail, some information on the functioning of modern operating systems is necessary, especially memory management and related terms such as programs, processes, functions, procedures are of great importance.

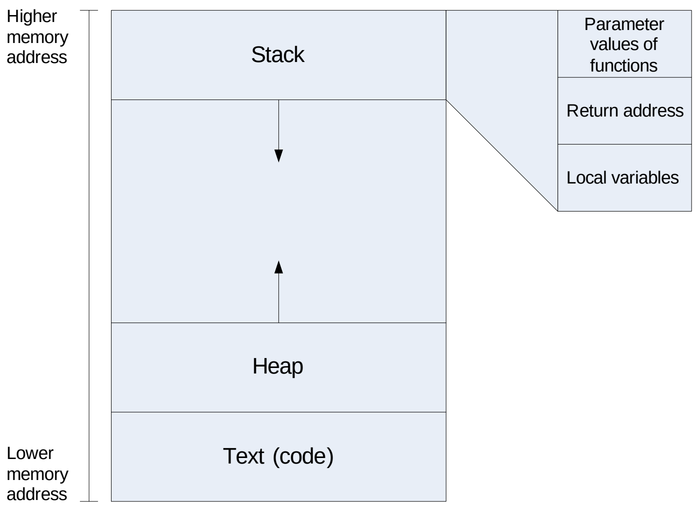


Fig. 12.2-7: Process in memory

Generally a process consists of three different parts: the heap, the stack and the data part (Fig. 12.2-7). The data part is the part where the program code is stored. Since the program code does not change at execution, this part is static and most importantly read only. In the heap, all data which is used program wide for example constants and global variables is stored. The heap is dynamic, but does not grow so fast. The third part of the process is called the stack. The stack grows from the top (higher addresses) to the bottom (lower addresses) and contains all dynamic data which is needed during the execution of procedures or functions, e.g. parameter values of functions, local variables, register values and most importantly the return address is stored here. The storage order is shown in Fig. 12.2-7 as well: at the top parameter values of functions, then the return address (Program Counter) and afterwards local variables. Since many values are stored and among them the return address, overflowing or manipulating the buffer to change the return address can lead to a serious security threat which will be addressed next using a conceptual example.

Example:
As can be seen from (Fig. 12.2-8 left side) *Program bufferoverflow* is the main program and consists of several instructions and a subroutine *buffer*. The addresses of the instructions are shown as well. The subroutine has the address n and the instruction where the program continues after the subroutine has been executed has the address $n + 1$. As can be seen from the stack 4 bytes are reserved each for the local variables x, y and z and another 4 bytes for the return address. When the subroutine *buffer* is invoked (Instruction n), the return address (the next instruction to be executed), in this case $n + 1$ is stored in the stack, then the local variable x, y and z respectively. Since the buffer grows from bottom to top, if we put more than the foreseen 4 bytes into the buffer reserved for z, the local variable y, the local variable x and most inportantly the return address $(n + 1)$ can be overwritten and replaced by a new return address.

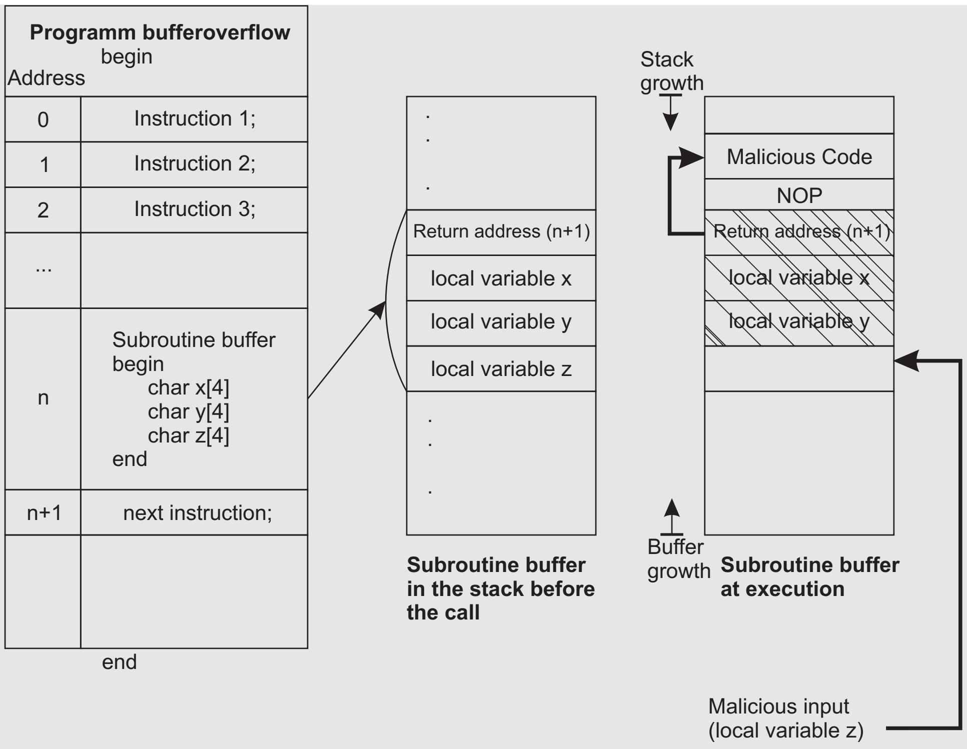


Fig. 12.2-8: Buffer overflow

If the content of the return address is overwritten, the program crashes if there is no reasonable code at the given new return address (segmentation violation under Unix), but if a useful code e.g. /bin/bash is available at the *new return address* the process will continue at the new return address and provide the attacker with a root shell if the crashed process was running with root privileges. Now that it has been shown how a buffer overflow is generated, the next question to be answered is how to get a useful code at the new return address. This can be done either by using code which is already available in the memory or by planting useful code (malicious input see Fig. 12.2-8) at the return address [Cri00]. Since the address of the executable code can not be exactly determined `No Operation(NOP)` are inserted. At the execution the `NOP` part will be ignored and the attack code executed.

Buffer overflow is a serious security threat, but can be prevented if diverse security measures are taken into account. The prevention can be carried out by for example checking the bounds of the inputs variables before they are processed, by writing better code (hand inspection), by analyzing the source code and by rendering the stack not executable ([Ist01]). In the next section we will deal with one of the most important and dangerous attacks (malware).

12.2.3 Malware

Malware related attacks are reported approximately every day. Nearly each computer user has to deal with this type of attack consciously or not. The costs related to malware are enormous: alone the damages caused by the *Love Letter worm* were estimated to some 5 billion Dollars. In this course unit we will deal with viruses, worms and trojans.

12.2.3.1 Viruses

According to Fred Cohen a computer virus is a program that can infect other programs by modifying them to include a possibly evolved version of itself [Coh85]. A virus can also be defined as a computer program that reproduces its own code by attaching itself to other executable files in such a way that the virus code is executed when the infected file is executed. In the definition of a virus there are two keywords: executability and replication, which will be looked at later.

The main structure of a virus is shown in Fig. 12.2-9. According to which a virus generally consists of four main parts ([Eck06]).

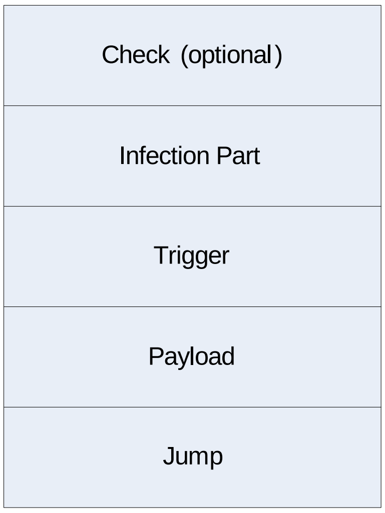


Fig. 12.2-9: Virus structure

The first thing each virus does is to infect as much executable files as possible by copying itself on a host program. But in order to work effectively the virus first checks (the checking is optional) whether the executable file that it wants to infect has already been infected or not. If the file has already been infected, the virus will not infect it again and will further proceed and look for not infected files. After the infection the virus may cause some damage such as deleting files, manipulating or altering data or devouring resources, etc. The damage is caused by the so-called payload and usually depends on an event (trigger). The event can be a calendar date, or a user input, etc. After the virus code has been executed there is a jump to the host code.

In the following section we will take a closer look at the most common types of viruses.

File Viruses

File viruses modify the content of COM- and EXE (and further files, which can be executed such as DLLs). They append the virus code either to the beginning or to the end of a file. The original file stays as it was. When the file is run, the virus code is executed first and there is a jump to the program code, the program code proceeds as if nothing has happened, the same as before the infection (see Fig. 12.2-10).

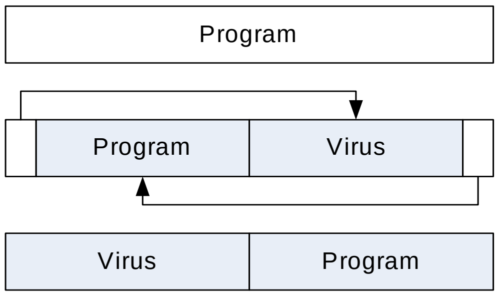


Fig. 12.2-10: File viruses

Terminate Stay Resident (TSR viruses)

Once executed TSR viruses stay resident in the memory (RAM) and manipulate the operating system so that, each time a program is started it is infected. If the PC is shut down, the virus is deleted from the RAM but the execution of a file which has been infected is enough to restore the previous state.

Stealth Viruses

Stealth viruses are a specific group of TSR viruses. Antivirus programs generate requests to look at system files which have been altered or manipulated. Stealth viruses intercept the result and send it back to the antivirus program. The result claims to the antivirus program that the state of the file is still the same as before the infection. Stealth viruses can intercept system calls and then generate their own answer.

Encrypted viruses

Encrypted viruses consist of an encrypted payload and an encryption routine. One of the most important characteristics is that for each encryption a different encryption key is used to make the virus look different. Therefore no signature can be found in the virus payload, but in the encryption routine, which always looks the same.

Polymorphic Viruses

Mutation engine

Polymorphic viruses work as encrypted viruses, but they have a further component, the so-called **mutation engine**, which changes the encryption routine after each infection and cleares the drawback of encrypted viruses. A polymorphic virus works as follows: when a file which has been infected by a polymorphic virus is executed, the encryption routine is run first and gains control of the further program execution. The virus payload and the mutation engine are decrypted, at this stage the decrypted virus takes control over program execution. The virus creates a copy of itself and of the mutation engine in the memory. The mutation engine generates a new encryption routine, which has the same functionality as the previous, but looks different. Afterwards the encryption routine encrypts the virus payload and the mutation engine and infects new files.

12.2.3.1.1 Macro viruses

Many applications (MS Word, MS Excel) have a very powerful macro library in order to support the user and ease the use of applications. But since macros are executable, all applications which have integrated macro functions can be infected by viruses. Through the use of an Auto-Open function the virus is executed when an Excel or a Word document is opened. Macro viruses are generally spread through e-mail with file attachments. A double click on the attachment suffices and the virus is run.

12.2.3.2 Worms

At first glance, viruses and worms have quite a few things in common, but if we take a closer look we will see that the differences are quite important. As has previously been said, generally a virus first checks if the program it wants to infect has already been infected to avoid double infection. Unlike viruses, worms are self replicating programs and do not infect any programs. Both, worms and viruses replicate, but the manner of replication is quite different. Viruses replicate by infection (modifying existing code) and by waiting for the host code to be run. Worms replicate by generating a copy of the worm program and proliferate by spreading over networks. To proliferate, the worm can either inject the worm code in the stack or overflow the buffer and run the worm application locally in the system. Another way used by many worms, to assure proliferation, is by sending itself as an e-mail attachment.

As the following examples show, worms are a great security threat and damages caused by them can range from simple jokes like displaying messages on the users screen to more serious things such as stealing passwords, deleting files or disrupting hosts from the internet by causing a DoS. There are also combinations of viruses and worms.

W32.MiMail.j@mm

Example 12.2-4:
W32.MiMail.j@mm is a worm, first published in 17.11.2003, which is sent as an attachment. The e-mail which is sent, gives the impression that it comes from a trustworthy person. When the attachment is opened, a form is shown, which requests credit card information. This information is then stored and sent to precise e-mail addresses. One of these attachments is `InfoUpdate.exe`. If the attachment is executed, the worm creates an entry in the registry. So that each time the computer is started, the worm is started as well.

Code Red worm

Example 12.2-5:
The "Code Red" worm was first published in july 2001. The worm provoked a buffer overflow on Internet Information Server II (version 4.0 and 5.0) running under the operating sytems Windows NT 4.0 or Windows 2000. To provoke the buffer overflow the worm generates an HTTP request on port 80, overflows the buffer and injects the worm code into memory as seen in Section 12.2.2.2. After overflowing the buffer the worm is run locally on the server and displays the message `"HELLO! Welcome to http://www.worm.com! Hacked By Chinese!"` which is why it is known as the Code Red worm. There were two versions of the Code Red worm. In order to proliferate the first version of the Code Red worm has an IP random number generator which generates IP addresses of hosts to be infected. But because of the static seed of this IP random number generator, only the same hosts were infectable. The second version implements a dynamic IP random number generator with the result that more than 359.000 machines were infected in just 14 hours. The damage caused by the worm depends on the date (trigger): from 1-19 of each month, the worm infects and proliferates; from the 20-27 of each month a Distributed Denial of Service (DDoS)[115] attack is performed on a fixed IP, from the 28-end of each month the worm sleeps.

12.2.3.3 Trojans

A trojan is a secret implementation of a set of instructions in a program (insertion of program chuncks without changing the documented task) or the secret installation of a program on a system. Characteristic for trojans is, therefore, that they do something else other than that what they are supposed to do.

Trojans can be active in many layers: implemented in an integrated development environment (IDE) they can copy themselves into all programs which are created

115 The idea behind DDoS is to make the attack more powerful by increasing the number of computers performing the attack, in the Code Red worm case a great number of infected computers participated in an attack on the White House IP address.

and operate in the firmware, operating system, service routines, database systems, etc.

The implementation of a trojan generally presupposes intensified insider knowledge over the system, otherwise the execution of the trojan will lead to an unexpected behaviour of the program and thus, to the recognition of the trojan. In programs which are very modular, the implementation of a trojan is easier.

Since the real processes which are carried out by the trojan are unknown to the user, the damages caused by trojans can be serious. A trojan code may be stealth as a game or a slightly harmful animation on the screen but the real process which is carried out without the users knowledge can be for example stealing passwords, spying, etc.

12.3 Intrusion Detection

In the following section we will deal with intrusion detection in greater detail. In this regard many questions have to be answered, e.g. what is an intrusion? What does the basic architecture for an intrusion detection system look like? What is the overall goal of an IDS? Which intrusions can be detected and most importantly what types of intrusion detection categories are available and what are their limitations. We will start by investigating which intrusion detection types are available and which approaches can be used to detect intrusions.

12.3.1 Intrusion Detection: Basic Principles and Concepts

According to Anderson 1980, who laid the cornerstone in the area of detecting intrusions, an intrusion can be viewed as a set of actions that attempt to compromise the integrity, confidentiality, or availability of computing resources via DoS, creating a back door (trojan horses), planting viruses and exploiting software vulnerabilities [An80]. Seven years later Denning 1987 defined an IDS as a software with the functions of detecting, identifying and responding to unauthorized or abnormal activities in a target system [Den87]. When dealing with intrusion detection, the overall goal to achieve is to make sure that the security policy is respected. A violation of the security policy can and will be seen as an intrusion [Kum95]. In line with this, intrusion detection can be viewed as detecting all activities which violate a predefined security policy. In Fig. 12.3-1 a general concept of an IDS is shown [Amo99]. A sensor monitors the target system and each time an activity happens which can be interpreted by the sensor as an intrusion, a report is generated to alarm the security officer responsible for the system. There is evidence which shows that traditional security methods such as firewalls (packet filters, proxies) and access control can not solve the problem we are facing today [Esc98].

Firewalls and ACLs

As the name already suggests, packet filters are used to filter packets based on the information contained in the header, they therefore operate at OSI layer 3 and 4 and do not have access to the content of packets. This means that all application related attacks can neither be prevented nor detected. Application gateways, commonly known as proxies operate at the application level (ISO layer 5 to 7), they can indeed access the data stream and filter it, but they have some drawbacks. Firstly, since the applications generally differ from each other, one needs a proxy for each application. Further, proxies are only available for the applications mostly in use (e.g. HTTP, SMTP, FTP, Telnet), this means that many other applications can not be protected through the use of proxies. Furthermore, they cannot always find out, whether the data which is transmitted is intrusive or not. A good example for the limitation of a proxy is the so-called *directory traversal attack* (see Example 12.3-1) on the Microsoft Internet Information Sever (IIS 3.0, 4.0 and 5.0), where through the use of unicode, directories could be accessed which are normally not accessible [Spe03].

Another major drawback of firewalls is that they can neither prevent attacks on protected areas nor new attacks.

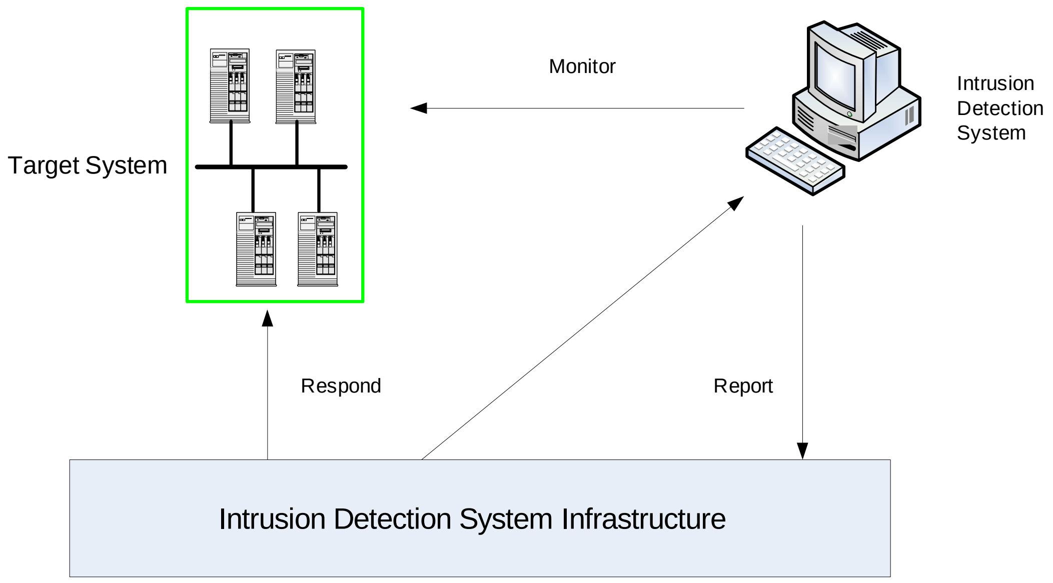


Fig. 12.3-1: Intrusion detection system

In environments where authentication and Access Control Lists (ACLs) are used to protect the system, user impersonalisation[116] as an example can not be detected. The limit of traditional security mechanisms has been reached here, more modern security methods are needed to address this issue and this is where intrusion detection starts to play a role.

116 E.g. the opponent Bob has sucessfully guessed the password of user Alice, and he registers as Alice.

12.3.2 Intrusion Detection Types and Approaches

In Fig. 12.3-2 the basic architecture of an IDS is illustrated and generally consists of four units. Depending on the way the data to be analysed is collected, we speak of Host-based Intrusion Systems (HIDSs) or of Network-based Intrusion Detection Systems (NIDSs). After the data has been collected it has to be processed to generate a format which is suitable for analysis. Diverse techniques can be used to analyse the data, some of which will be explored in more detail in this course unit. In case an intrusion is discovered an alarm is sent to the security officer who then takes appropriate measures. At the beginning of this section it was mentioned that sensors can either be installed on a single host to collect data (HIDS) or network wide (NIDS) to monitor whole network segments. In the following we will take a closer look at both, analyse the differences, and highlight the advantages and disadvantages of each philosophy, starting with HIDS.

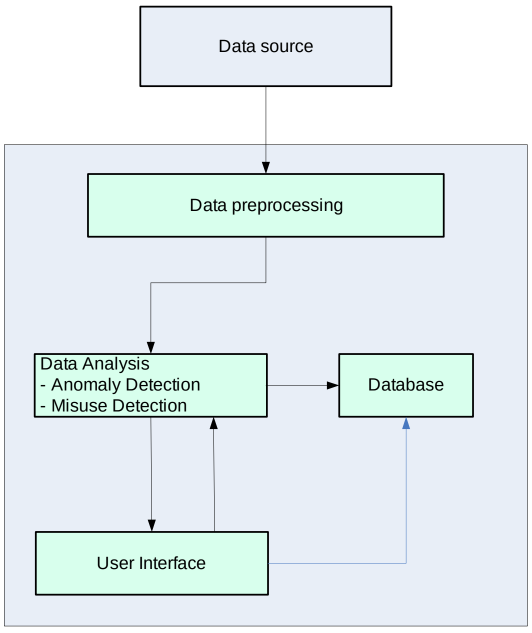


Fig. 12.3-2: IDS: basic architecture

12.3.2.1 Host-based Intrusion Detection

In case of a host-based IDS, the software is installed on each host. Diverse methods can be used to detect intrusions on a host, some of which (Log File Analysis, and System Integrity Verification) will be dealt with in greater detail.

12.3.2.1.1 Analysis of Log Files

Generally, on each computer system diverse events (file access, access to system resources, unusual behaviour, etc.) are recorded and stored in the so-called *log files*. Carefully analysed, these log files can be used to detect intruders. On Windows, three types of events are recorded (application related events, system related events and most importantly security related events). In Solaris the so-called *Basic Security Module* (BSM) records all security relevant events, nearly 250 security relevant events are available. In the past the BSM has been used in many studies to detect intrusions [Kor93][Por92]. In Unix, diverse log files are used to record the activities on the system e.g. syslog[117], lastlog[118], faillog[119]. In the following we will provide some examples of how log files can be used to detect intruders.

BSM

Directory traversal attack

Example 12.3-1:
Assume we have a web server running on a host, how can we use log file analysis to detect attacks on the web server? In the case of a web server, two log files are available, access log and error log. An abstract of access log is shown below:

```
132.176.12.173 [06/Jun/2006:10:30:16 +0100]
GET /middle.html HTTP/1.1  200 478

132.176.12.173 [06/Jun/2006:10:35:14 +0100]
GET /alice.gif HTTP/1.1 304

132.176.12.173 [06/Jun/2006:10:37:14 +0100]
GET /home/user/down.html HTTP/1.1 200 367

132.176.12.173 [06/Jun/2006:10:39:14 +0100]
GET /print.gif HTTP/1.1 304

132.176.12.173 [06/Jun/2006:10:42:14 +0100]
GET /logo.jpg HTTP/1.1 404 297

62.104.86.112  [06/Jun/2006:10:47:19 +0100]
GET /scripts/.\%252e/.\%252e\/winnt/system32/cmd.exe?/
c+dir+c: HTTP/1.1\ 404 325

151.198.253.35 [06/Jun/2006:13:01:46 +0100]
GET /scripts/  .\%\%..\%255c\%255c../winnt/system32/cmd.exe?
/c+dir 404 -

211.81.24.3    [06/Jun/2006:15:20:34 +0100]
CONNECT 1.3.3.7:1337 HTTP/1.0 404 273
```

117 Syslog records all system calls.

118 Lastlog records last logins.

119 Here all failed logins are stored.

As can be seen from the log file each log entry consists of the IP address of the client who accesses the server, the date and time the file has been accessed, followed by the method used to invoke the resource, the resource the client has requested, the current version of the HTTP protocol, and finally a code to indicate what happens. The common one are 200 (`OK`) and 404 (`Not Found`). In the first row of this log file, the code number `200` tells us that the client with the IP number `132.176.12.173` has sucessfully accessed the resource `middle.html`. A closer look at other rows, e.g. those starting with the IP addresses `62.104.86.112` and `151.198.253.35`, clearly shows us that there has been an attempt to perform a directory traversal on the server. Normally a web server only allows users access to web folders and related subfolders. But if the web server has a bug, further folders (even those above the root directory) can be accessed and applications could be invoked. As seen in this example the users with the IP addresses `62.104.86.112` and `151.198.253.35` try to execute the command `cmd.exe?/c+dir+c` to display a list of files in the `C:\` directory.

Every page which can not be accessed whatever the reason has a protocol of this in its error log as the following example shows.

Example 12.3-2:

```
217.238.141.213 [12/Jun/2006:09:04:13 +0100]
GET /main.php HTTP/1.0 404 281

217.238.141.213 [12/Jun/2006:09:06:13 +0100]
GET /phpinfo.php HTTP/1.0 404 284

217.238.141.213 [12/Jun/2006:09:08:13 +0100]
GET /test.php HTTP/1.0 404 281

217.238.141.213 [12/Jun/2006:09:14:14 +0100]
GET /index.php3 HTTP/1.0 404 283

128.206.132.141 [12/Jun/2004:10:12:16 +0100]
GET /scripts/..\%255c\%255c../winnt/system32/cmd.exe?/c+dir" 404

218.61.34.188   [14/Jun/2006:14:41:42 +0100]
GET /d/winnt/system32/cmd.exe?/c+dir   404
```

In the first row, the code 404 tell us that there was an attempt to retrieve the web site `main.php`, but unfortunately an error has occurred since the web site is no longer available, most web users will be familiar with code 404. The two last entries of the error log table show that there was, as in the case of access log, one more attempt to perform a directory traversal `(GET /scripts/..\%255c\%255c../winnt/system32/`

`cmd.exe?/c+dir)`. Since the IP number of the client who has tried to browse the `C:\` directory is known, it can be used to take further steps (under the assumption the IP address has not been spoofed).

Unlike access log and error log which are used to protocol web servers accesses and errors, lastlog is a standard Unix log file which shows us who last accessed the host.

Example 12.3-3:
Below an abstract of the last log file is shown.

```
Jun 14 11:55:02 forst sshd: Failed password for root from
132.176.12.123 port 1137 ssh2

Jun 14 11:55:12 forst sshd: Failed password for root from
132.176.12.123 port 1137 ssh2

Jun 14 11:55:23 forst sshd: subsystem request for sftp

Jun 14 11:55:34 forst sshd: Failed password for root from
132.176.12.144 port 1139 ssh2

Jun 14 11:55:45 forst sshd: subsystem request

Jun 14 11:55:56 forst sshd: Failed password for root from
132.176.12.129 port 1135 ssh2
```

The abstract clearly reveals to us, that the user with the IP address 132.176.12.123 has tried to access the system as root. But the registration failed twice. This can not immediately be interpreted as an attack. But if a rule is created by the HIDS to record the number of failed attempts within a period of time, an alarm can be automatically generated if a threshold, which has been defined, is reached.

Using the lastlog file it is also possible to identify unusual behaviour, e.g. if we clearly know that Alice usually last logs at the latest 5pm, suddenly a last login, according to which, she had last logged at 3am would be extremly suspicious. Logsurfer and many other HIDSs based on log file analysis, work exactly according to the principle described above.

12.3.2.1.2 System Integrity Verification

Another method which is often used in HIDS is **System Integrity Verification**. **SIV** Here, a snapshot of a system is made at some stage. All system files which are relevant for the security of the system are recorded and stored in a database or in a file. At certain time intervals, the actual system state is recorded and compared to the system state stored in the database. Since system data generally does not change, a deviation or a difference between both system states denotes an intrusion. This method is straightforward but it has to be ensured, that at the moment the snapshot of the system was taken, the system was not compromised and that the database file where the system state is stored is write protected or protected against intruders.

Tripwire

Example 12.3-4:
One of the most often used system integrity verifier is Tripwire. Tripwire uses cryptographic check sums to test the integrity of systems files. The fingerprint of each file is made and stored in the database. If a file is modified by a hacker, it will provide a different fingerprint. Here there is no possibility to hide the modification by modifying a file but making sure that the file attributes which are controlled such as date, file length etc. remain the same. Tripwire can detect such modifications, since the fingerprint will differ from that of the snapshot.

Host-based IDSs have many advantages: log files are available which clearly provide us with a lot of different information e.g. files that have been accessed, access failure, user logins, or resource usage. In short, very precise information is available about what happens on the host. Further, since the software is installed on a host, it is independent of the network speed. HIDSs have their drawbacks as well. Since the software has to be installed on several hosts, more work can be involved than is required for the installation of NIDSs. Further HIDSs are dependent on the operating system they are installed on.

HIDSs can detect all host intrusions such as file modification, user impersonalisation, password guessing, attacks with encrypted traffic, etc.

12.3.2.2 Network-based Intrusion Detection

As previously mentioned, the data source is the starting point of an IDS. In networks, network packets are analysed to detect intrusions. Hereby the header and payload of each packet traversing the network are recorded and statistical methods are used to detect unusual behaviour (Anomaly Detection) or rules are defined according to which the analyser has to deal with the packets. Some of the important questions which arise at this stage are where to place the sensor to collect the packets and how to deal with fragmented packets.

NIDS placement

Depending on the topology of the network the NIDS can either be placed at the front of or behind the firewall. Placing the NIDS in front of the firewall has some advantages. All external traffic traversing the network can be monitored and theoretically all intrusions which can be seen, and most importantly, attacks, also those against the firewall can be detected. But placing the sensor in front of the firewall has one enormous drawback, internal intruders, who according to various statistics are a great problem, can not be detected at all. On the other hand by placing the sensor behind the firewall, internal intruders who are misusing the system can be detected but since undesired traffic is blocked by the firewall, it is not possible to see all attempted intrusions. Further, attacks against the firewall as such can not be detected. Therefore it is recommended to use both, an internal and an external intrusion detection system (Fig. 12.3-3).

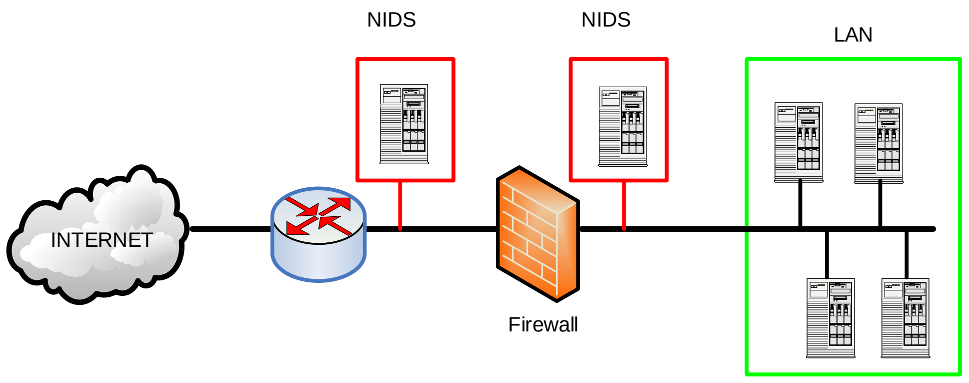


Fig. 12.3-3: NIDS placement

As we have seen at the beginning a NIDS analyses each packet which enters the network segment it is installed on. But, a NIDS which only looks for known patterns in single packets, a so-called *packet analyser* [Spe03] will not be able to detect attacks which are spawn over multiple packets. Therefore a good intrusion detection system should be able to deal with the fragmentation and reassembly of packets.

Fragmentation and reassembly

Compared to HIDSs, the great advantage of NIDSs is that they are independent of the underlying operating system and therefore portable. Since the software operates network wide, there can be less work involved in managing them than HIDSs. NIDS have some drawbacks as well, current NIDSs can barely handle Gbps traffic and because many companies are now installing lines with Gbps, NIDSs may have problems to follow. Further, they are blind to encrypted traffic and have a relatively high rate of false positives[120] compared to HIDSs. Intrusions which can be detected by NIDSs are: spoofing, DoS attacks, buffer overflows, port scanning etc.

12.4 Major Intrusion Detection Modelling Techniques

As previously mentioned detecting anomalies means constructing a subject profile. To construct these profiles diverse techniques can be used, e.g. statistical models [Mar01], data mining models [Wen98], neural networks [Rya98], decision trees, etc. To detect misuse, models such as state transition analysis [Nstat][Por92][Kor93] and rule based misuse detection are used. We will start by taking a closer look at anomaly detection.

120 False positive are alarms generated although there was no intrusion.

12.4.1 Anomaly Detection: Statistical Approaches

By the anomaly detection a typical system behaviour has to be built first. This means that in the preliminary stage of building the system a lot of measures have to be taken to determine the normal or typical behaviour of the system or users of the system, the so-called reference value of the user behaviour or system. This is a very difficult task: firstly because it is difficult to determine what is normal and secondly because users who know that their profile is being made can consciously behave differently to normal in order to falsify their profile. To build up the normal behaviour, often attributes relating to event counter (number of false logins within a period), resource usage (CPU usage), session duration, user last logins, etc. are used. If we assume a time discrete model, generally we have n observations $x_1 \ldots x_n$ and the goal is to check whether the next observation x_{n+1} is abnormal or not [Den87]. In the following section we will see how statistical models, e.g. mean and standard deviation model and the Hotelling´s test [Ye01][Ye02][Ye01a][Sac93] can be used to detect intruders.

12.4.1.1 Univariate Analysis: Mean and Standard Deviation Model

Assume $x_1 \ldots x_n$ is an observation, then the mean of this observation can be determined and equals

$$\bar{X} = \sum_i \frac{x_i}{n}, \qquad 12.4\text{-}1$$

with x_i the value of the the variable X at time i. The standard deviation σ can be defined as well

$$s^2 = \frac{\sum_{i=1}^{i=n}(x_i - \bar{X})^2}{n-1} \quad with \quad \sigma = \sqrt{s^2}. \qquad 12.4\text{-}2$$

A new observation is classified as an intrusion if it does not fit into predefined boundaries or into a tolerance interval. Using the mean and standard deviation model, the procedure to detect intrusions is summarized below:

1. The activity profile of a user, e.g. the CPU usage is recorded and sampled, we obtain $x_1 \ldots x_n$ with x_i the value of the profile at time i.

2. The mean and the variance of the activity is determined according to Eq. 12.4-1 and Eq. 12.4-2.

3. Now, when the system records the next sample x_{n+1} it has to decide whether an anomaly has occurred or not. Here the boundaries within which the value x_{n+1} has to be in order to be seen as normal can be determined by using the *Chebyshev´s Inequality*.

$$(P(|X - E(X)|) \quad \leq \epsilon) \geq 1 - \frac{\sigma^2}{\epsilon^2} \qquad 12.4\text{-}3$$

The **Chebyshev´s Inequality** can be used independently of the form of the curve. In the special case of a normal distribution for the probability density of the subject profiles, the next sample can be classified as being normal or abnormal using the theoretical sample distribution of the mean (Fig. 12.4-1). It is already known that, approximately

Chebyshev´s Inequality

Theoretical sample distribution of the mean

- $68,27\%$ of the samples fall between $\mu \pm \cdot\sigma$;
- 95,45 $\%$ between $\mu \pm 2 \cdot \sigma$;
- and 99, 73 $\%$ between $\mu \pm 3 \cdot \sigma$ [Sac93][Spi90].

If we want to detect intruders using the normal distribution we will first have to determine the level of accuracy. By choosing a confidence level of $0,9973$, we want $99,73\%$ of all our samples to be between $\mu \pm 3 \cdot \sigma$. In this case, each time a sample x_{n+1} is observed, x_{n+1} has to lie in the intervall $\mu \pm 3 \cdot \sigma$ (three standard deviations away from the mean), otherwise there is an intrusion.

This model has a major drawback, only one variable is taken into consideration. But it is known that intrusion activities are generally correlated. In other words, when an intrusion happens, the probability that more that one variable is involved is large. This has led to methods which take this into account, called the *Multivariate Analysis*.

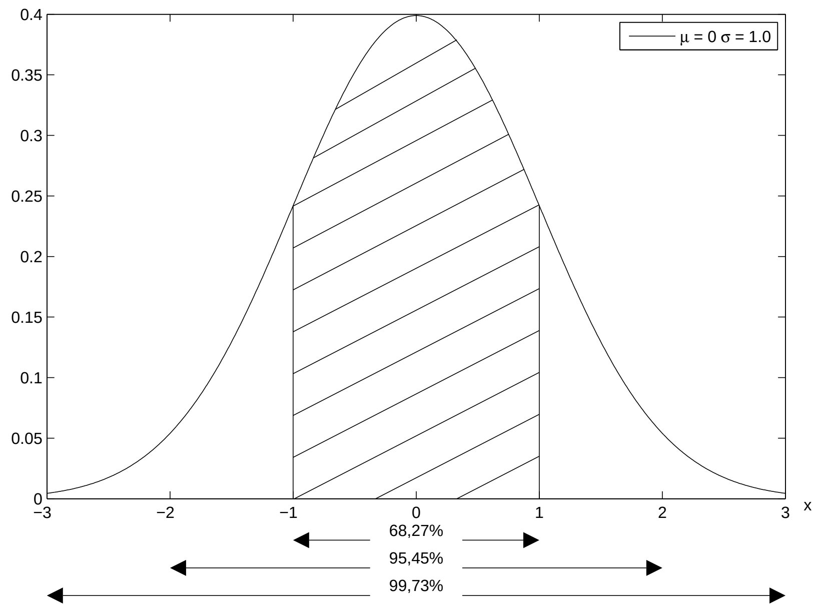


Fig. 12.4-1: Theoretical sample distribution of the mean.

12.4.1.2 Multivariate Analysis

In the multivariate analysis more than one variable is analysed and correlations between variables are included in the model. In this case instead of monitoring and analysing only the CPU usage, the number of file accesses and the number of logins can be taken into consideration as well. There are diverse methods (Hotelling's test, Chi-square test see Appendix A.2) which can be used to analyse multiple variables, in this course unit we will focus on the Hotelling´s test.

In the case of univariate analysis, we have proceeded as follows: at first we have calculated the mean and the variance of a variable. Using the Chebyshev´s Inequality or the theoretical sample distribution of the mean we were able to fix the boundary and make a statement about intrusion or not. In the case of multivariate analysis we will proceed in the same way. Firstly we determine the mean of the multivariable. Let us denote X this variable, then $\bar{X} = [\bar{X}_1, \bar{X}_2, \bar{X}_3, \bar{X}_4, \ldots \bar{X}_p]$ where $\bar{X}_i$ is the mean of the variable X_i. Further, as in the univariate case we will determine the variance of the multivariable X, which is the variance of each variable taken on its own and since there can be correlations between variables, the covariance between each pair of variables. The result is the so-called *variance-covariance matrix.*

Variance-covariance matrix

Be M a matrix of observations with p being the number of variables to be analysed and n the number of observations for each variable. M can be written as follows

$$M = \begin{bmatrix} \mathtt{x}_{11} & \mathtt{x}_{12} & \mathtt{x}_{13} & \ldots & \mathtt{x}_{1p} \\ \mathtt{x}_{21} & \mathtt{x}_{22} & \mathtt{x}_{23} & \ldots & \mathtt{x}_{2p} \\ \mathtt{x}_{31} & \mathtt{x}_{32} & \mathtt{x}_{33} & \ldots & \mathtt{x}_{3p} \\ \ldots & & & & \\ \mathtt{x}_{n1} & \mathtt{x}_{n2} & \mathtt{x}_{n3} & \ldots & \mathtt{x}_{np} \end{bmatrix} \qquad 12.4\text{-}4$$

The variance-covariance matrix S can be calculated as follows [Ye01]

$$S = \frac{1}{(n-1)} \cdot \sum_{i=1}^{i=n} (X_i - \bar{X})(X_i - \bar{X})^t, \qquad 12.4\text{-}5$$

Hereby X_i is a row vector of the matrix M.

The Hotelling´s test for a new observation X_e is

$$T^2 = (X_e - \bar{X})^t \cdot S^{-1}(X_e - \bar{X}) \qquad 12.4\text{-}6$$

As we can guess, Eq. 12.4-5 has the same structure as in the univariate case. Now that it has been shown how the mean and the variance-covariance matrix are calculated, we will proceed further and determine the condition under which a next observation $X_{n+1} = x_{n+11}, x_{n+12}, \ldots x_{n+1p}$ can be classified as normal or abnormal. To determine whether a new observation X_{n+1} is normal or abnormal, we apply the Hotelling´s test as follows:

Hotelling´s test

1. Determine the mean $\bar{X}$ of the variable X
2. Determine the variance-covariance matrix S according to Eq. 12.4-5
3. Compute the Hotelling´s test T^2 for the event X_{n+1} according to Eq. 12.4-6.

A new observation is abnormal if the computed value of T^2 depasses an upper limit. To determine this upper limit the Hotelling´s T^2 statistic has to be transformed into an F-distribution (Appendix A.3) with p and $(n-p)$ degrees of freedom [Ye01a]. This is achieved by multiplying T^2 with a constant $\frac{n\cdot(n-p)}{p\cdot(n+1)\cdot(n-1)}$. The obtained value $T^2\cdot\frac{n\cdot(n-p)}{p\cdot(n+1)\cdot(n-1)}$ is then compared to the tabulated F value for a given level of significance α. If the calculated value is greater than the tabulated value (critical value) it will then be assumed that an intrusion has occurred.

Level of significance α

Example 12.4-1:
The result of an experiment involving three variables (X_1, X_2, X_3) is shown below

$X_1 = (1,2 \quad 1,4 \quad 1,3 \quad 1,5 \quad 1,2)$, $X_2 = (3,1 \quad 3,4 \quad 3,3 \quad 3,6 \quad 3,4)$, $X_3 = (5,4 \quad 5,2 \quad 5,3 \quad 5,0 \quad 5,5)$. We have 5 observations for each variable, and it has to be checked whether the next observation $(1,3 \quad 3,37 \quad 5,23)$ is an intrusion or not. In order to do this, we will first have to write down the observations as a matrix

$$M = \begin{bmatrix} 1,2 & 3,1 & 5,4 \\ 1,4 & 3,4 & 5,2 \\ 1,3 & 3,3 & 5,3 \\ 1,5 & 3,6 & 5,0 \\ 1,2 & 3,4 & 5,5 \end{bmatrix}$$

Now we have to determine the mean vector, by calculating the mean value of each variable $\bar{X} = (1,32 \quad 3,36 \quad 5,28)$. We can determine the variance-covariance matrix S according to equation Eq. 12.4-5. Here X_i is a row vector of the matrix M. We will determine the first factor in the equation Eq. 12.4-5. $X_1 = (1,2 \quad 3,1 \quad 5,4)$ and $(X_1 - \bar{X})\cdot(X_1 - \bar{X})^t =$

$$\begin{pmatrix} -0,12 \\ -0,26 \\ 0.12 \end{pmatrix}\cdot(-0,12 \;\; -0,26 \;\; 0,12) = \begin{pmatrix} 0,0144 & 0,0312 & -0,0144 \\ 0,0312 & 0,0676 & -0,0312 \\ -0,0144 & -0,0312 & 0,0144 \end{pmatrix}$$

The same is done for all other factors and put in Eq. 12.4-5. We will not carry out the operation here in detail, the result is shown in the matrix

$$S = \begin{bmatrix} 0,017 & 0,0185 & -0,0245 \\ 0,0185 & 0,033 & -0,0235 \\ -0,0245 & -0,0235 & 0,037 \end{bmatrix}$$

The interpretation of the results clearly shows that Cov (X_1, X_3) = -0,0245 this means that the variables X_1 and the X_3 are not dependent on each other, but since Cov (X_1, X_2) is positive X_1 and X_2 are correlated. After determining the S matrix we can now calculate the Hotelling´s test T^2 for the new observation $(1,3 \quad 3,37 \quad 5,23)$. In this case T^2 equals

$$(-0,02 \quad 0,01 \quad -0,05) \cdot \begin{pmatrix} 0,017 & 0,0185 & -0,0245 \\ 0,0185 & 0,033 & -0,0235 \\ -0,0245 & -0,0235 & 0,037 \end{pmatrix}^{-1} \cdot \begin{pmatrix} -0,02 \\ 0,01 \\ -0,05 \end{pmatrix}$$

$$T^2 = 8,31$$

This value is then transformed to an F value by multiplying it with a constant $\frac{n\cdot(n-p)}{p\cdot(n+1)\cdot(n-1)}$ and then compared to the tabulated value (seehttp://www.socr.ucla.edu/Applets.dir/F_Table.html). $p = 3$ (3 variables) and $n = 5$ (5 observations for each variable). Therefore $T^2 \cdot \frac{n\cdot(n-p)}{p\cdot(n+1)\cdot(n-1)} = 1,154$.

The tabulated value for $p = 3$ and $n - p = 2$ for a level of significance $\alpha = 0.05$ is 19.1643. Since the calculated value is smaller than the tabulated one, there is no intrusion.

The Hotelling´s test has a big drawback, then to determine the variance-covariance matrix and its inverse is very time consuming and since in modern networks, attacks have to be detected in real time, a faster method is needed, this has led to another proposition for detection of outliers using the Chi-square test [Ye01a]. With the Chi-square test the matrix inversion is not necessary, only the distance from the mean vector is taken into account.

12.4.2 Misuse Detection

In misuse detection, another approach is pursued. It is assumed that whenever an intrusion happens, an intrusion specific signature is left. If this signature is analysed carefully, then suspicious attacks can be detected.

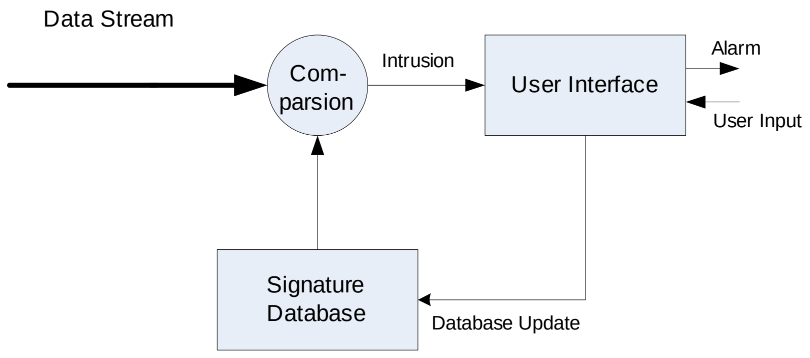


Fig. 12.4-2: Misuse detection

In Fig. 12.4-2 the principle is shown. A signature database of all known attacks is available. The data stream is then analysed and searched for known attacks stored in the database. If there is a match, an alarm is generated to inform the security officer. Whenever new attack signatures are available, the database has to be updated. Some of the approaches often used to detect misuse are state transition analysis and rule based misuse detection.

State transition analysis

In **state transition analysis**, a computer intrusion is seen as a sequence of actions leading from an initial limited acces, generally less powerful state (normal user) to a final compromised state, generally a more powerful state (root) ([Por92][Kor93]). As the pattern matching technique, state transition analysis is based on a predefined attack signature. The technique works as follows: firstly known attacks are modelled as state transition diagrams and stored in a database. Each time an event starts, e.g. a user logs in, the activities are recorded and a transition diagram relating to the activities is generated and compared to that of known attacks. If both diagrams match, there is an intrusion. If we take a closer look at the approach of modelling intrusions using state transition analysis, then we realise that since a state diagram consists of states and transitions between states, we firstly have to define what the states of our system are and most importantly we have to determine those events or key actions which lead to a transition from one state to another state.

12.4.2.1 Rule based Misuse Detection (Pattern Matching)

The general idea using this method is to build a signature database of known attacks and use rules to fire if known attacks are seen in packets. A normal three way handshake mechanism to initiate a TCP connection between sender and recipient has the following flags: SYN, SYN/ACK and ACK. A TCP half open scan in contrast only has one flag set and a series of SYN without acknowledgement, which can be used to perform a DoS on a server. Since this type of attacks are already known, rules can be generated to avoid them and alert the administrator in case packets with such patterns are seen. There are many NIDSs which work based on rules, a famous one is Snort.

Snort

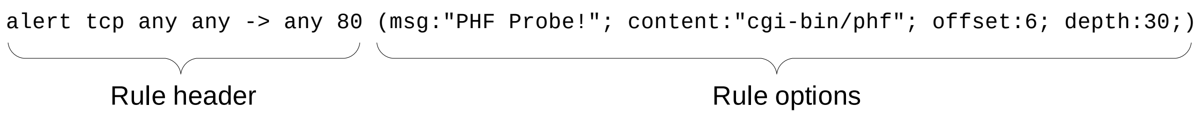
Snort is an open source tool which has been designed to protect small networks in the range of 100 Mbps. Using Snort three operational modis are available:

- sniffer
- packet logger
- and IDS.

If Snort has been configured as a sniffer, all packets traversing the network can be seen. In the packet logging mode, the packets are recorded on the disc. In the intrusion detection mode, Snort uses rules, which are defined in the configuration file `snort.conf` to detect intrusions.

The general anatomy of a snort rule is shown in Fig. 12.4-3 and consists of two parts:

- the rule header and
- the rule options.



```
alert tcp any any -> any 80 (msg:"PHF Probe!"; content:"cgi-bin/phf"; offset:6; depth:30;)
```

Fig. 12.4-3: Anatomy of a Snort rule

Snort analyses each packet traversing the network in two steps. At first the header of the Snort rule is applied to the packet. If there is no match, the packet is ignored, otherwise the rule options are applied, in case rule header and rule options match, actions are taken. In the following we will take a closer look at the anatomy of a Snort rule starting with the rule header.

Rule header

As can be seen from Table 12.4-1 the rule header describes the endpoints of the communication. Each rule header starts with the action to be taken in case the rule fires.

Tab. 12.4-1: *Rule header*

Action	Protocol	Source IP	Source Port	Direction	Destination IP	Destination Port
`alert`	`tcp`	`any`	`any`	$\rightarrow$	`any`	`80`

- **Action**

  Currently 5 rule actions are available:

  `alert:` in this case the packet is recorded and an alarm is generated. In the current version of Snort seven alert modes are available. In the default mode the packet header is printed together with the alert message. Snort can be configured as well to send an alert message to the screen.

  `log:` the packet is recorded, no alert message is generated.

  `pass:` tells Snort to ignore the packet.

  `activate` an alert message is generated and further so-called *dynamic rules* are activated.

  `dynamic` they are actived by previous rules.

- **Protocol**

  Here the protocols to be analysed for suspicious behaviour are specified. Currently Snort analyses 4 protocols (ICMP, IP, TCP and UDP). Further protocols such as ARP, IPX (Internet Packet eXchange), RIP (Routing Information Protocol), etc. will be supported in the near future.

- **Sender/Receiver IP address**

  The sender/receiver IP address in the CDIR (Classless Interdomain Routing, see RFC 1517) notation is used to identify the sender/receiver of the message. Snort allows the use of variables to define a range of addresses as well. A dollar in front shows that it is a variable and indicates the IP range to be observed (see Example 12.4-3). The variable must be defined in the configuration file `snort.conf`.

- **Sender/Receiver Port**

  Ports can be specified either as numbers or ranges. The number range is from 1 to 65535. Since port numbers are irrelevant in an IP packet, rule will be applied to all packets. Variables can be used to define port range as well.

- **Direction**

Indicates the direction of the traffic the rule should apply to (unidirectional or bidirectional).

If there is a match for the rule header, options are applied to the rest of the packet to further analyse it. Currently diverse options are available. Some of which will be introduced next.

Rule Options

As can be seen from Fig. 12.4-3 options are the second part of the Snort rule and are in parentheses. They tell Snort which attributes of the packet are to be checked further. The options start with a keyword followed by the option argument (the value of the option to be checked). Options and their arguments are separated with a colon and options are separated from each other by a semicolon.

Keyword	Separator	Argument	Separator	Further options

msg	:	"PHF Probe!"	;	content:"cgi-bin/phf";
				offset:6;
				depth:30;)

1. **IP Protocol options**
   - `sameip:` checks whether the sender and the receiver IP is the same (see Example 12.4-2) for the use of this option.
   - `TTL:` checks the Time-to-Live, it is useful for the detection of traceroute attempts.
   - `fragbits:` checks the fragment bits in the IP header. Three bits can be checked:

     `R-bit (reserved bit)` is reserved for future use.

     `D-bit (do not fragment)` in case this bit is set, the router is not allowed to fragment the packet.

     `M-bit` indicates that more fragments are coming. All packets except the last one have the `M-bit` set to 1.
2. **TCP protocol options**

   Attributes of the TCP header which are often analysed by Snort to look for attacks are flags. Since many attacks rely on crafted packets (misconfiguration of the flags to generate bad traffic) .

   - `Flags`: `F`, `S`, `R`, `P`, `A` and `U` bits or a combination of them is checked. The sequence number and acknowledment number can be checked as well.
3. **content**

   Content is one of the most powerful options, then in many cases such as buffer overflow or malware the content of the packet (payload) has to be analysed in order to be able to detect the attack. Some of the attributes used in relation with the content are described next.

- `content:` searches for specific patterns. Binary patterns can be searched for as well, they are enclosed in pipe (||).
- `uricontent:` tells Snort to only check the URI (Unified Resource Identifier) part of the packet.
- `offset:` normally the Snort engine searches for specific pattern in the whole payload, but with the offset option it can be specified where to begin to search.
- `depth:` the maximal depth to search for.
- `nocase:` not case sensitive.

4. **Meta-data rule options**

   - `msg:` message to be logged together with the packet to quickly identify the attack.
   - `reference:` to find further information related to the attack.

Currently Snort has a database with nearly 2200 predefined rules some of which are shown next.

Example 12.4-2:
A *land attack* consists of sending a crafted IP packet where the source address equals the destination address. On Computers running unpatched versions of Windows NT, the computer crashes if the sender address is the same as the receiver address. The rule to prevent such attacks is shown below:

```
alert ip any any -> any any (msg:"DOS Land
attack";sameip;)
```

Let us break down the rule:

- `alert:` indicates it is a highly sensible message (alert message)
- `ip` the IP protocol header is checked
- `any` from any IP address
- `any` from any port
- $\Rightarrow$ indicates the direction that has to be checked.
- `any` to any IP address
- `any` on any receiving port
- `msg` indicates which kind of attacks are dealt with, in this case a *DOS land attack*, the message is enclosed in brackets
- `sameip` this rules looks for spoofed packets with source IP = destination IP.

Example 12.4-3:

```
alert ip $EXTERNAL_NET  $SHELLCODE_PORTS -> $HOME_NET  any
(msg: SHELLCODE x86 NOOP;content: |90 90 90 90 90 90 90 90 90
 90 90 90 90 90|; depth: 128;).
```

This rule looks for attempts to overflow a buffer as seen in Section 12.2.2.2 and execute arbitrary code in the stack. Snort generates an alert whenever an IP packet is observed coming from `EXTERNAL_NET` headed to `HOME_NET` containing the binary string `"|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|"`. The pipes are there to indicate that this string is a binary one. The option `depth`: 128 specifies that only the first 128 bytes should be searched for.

With the open source tool Snort a great range of attacks can be detected (exploits, bad traffic, malware, etc.). Compared to the anomaly based method, the rule based approach has a relatively low amount of false positives. The greatest disadvantage of this method is that unknown attacks can not be detected. Further, the signature database has to be updated on a regular basis.

A Distributions

A.1 Normal Distribution

As can be seen from Eq. 1 the normal distribution can be fully characterized by two terms: the height μ (the mean of the variable x) and the width σ (the standard deviation). Assume we would like to detect intruders on a host based on CPU utilisation. Firstly we will have to determine the mean of the CPU usage, which will clearly show us the average CPU usage and then we will have to determine the CPU usage deviation from the mean. When we have these two values, we can specify our confidence interval, which is the interval we want our normal CPU usage to be within.

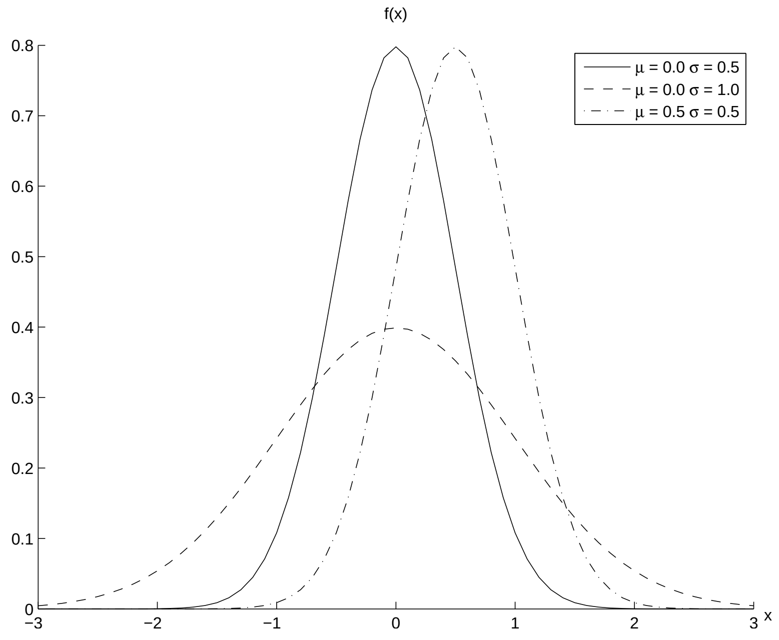


Fig. 1: Normal distribution

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \qquad 1$$

For many applications the probability of a variable X being between an interval a and b under the curve of the nornal distribution is of great interest, then, if we can define a confidence interval (where all *good values* lie) all values not belonging to this interval can be seen as anomalies.

The probability of a value X, being between a and b can be calculated and equals:

$$P(a \leq X \leq b) = \int_a^b f(x)dx \qquad 2$$

and the probability of a value being under the curve of the probability density function is:

$$P(-\infty \leq X \leq \infty) = \int_{-\infty}^{\infty} f(x)dx = 1 \qquad 3$$

In practice the probabilities of the variable X being between $\mu \pm \sigma$; $\mu \pm 2\sigma$; $\mu \pm 3\sigma$ is of great importance and has already been calculated (see Fig. 12.4-1):

$$\int_{\mu-\sigma}^{\mu+\sigma} f(x)dx = 0,6827; \int_{\mu-2\sigma}^{\mu+2\sigma} f(x)dx = 0,9545; \int_{\mu-3\sigma}^{\mu+3\sigma} f(x)dx = 0,9973 \qquad 4$$

A.2 The Chi-Square Distribution

Before we introduce the Chi-square distribution we would first like to introduce the Chi-square test. Normally when we carry out an experiment, we expect the result of the experiment to differ from the empirical result. But if we want to know to what extent the result of the experiment differs from the empirical result we can use the Chi-square test to find out.

The starting point of each Chi-square test is the so called *Zero Hypothesis* which relies on empirical results. When we toss a coin 10 times, we expect to get heads 5 times and tails 5 times. We therefore asssume that the coin is perfect (the coin is fair): this is our Zero Hypothesis. Depending on the result of the experiment we can either confirm the Zero Hypothesis or reject it. The formula used to calculate the Chi-square test is

$$\chi^2 = \frac{(O_1 - E_1)^2}{E_1} + \frac{(O_2 - E_2)^2}{E_2} + \ldots + \frac{(O_n - E_n)^2}{E_n} = \sum_i \frac{(O_i - E_i)^2}{E_i} \qquad 5$$

where E_i is the expected value based on empirical experiences and O_i is the oberved value.

As can be seen from Eq. 6 the probability density function (PDF) of the Chi-square variable only depends on one parameter, the so called
degree of freedom[1] $(\nu = n - 1)$.

$$f(x) = \frac{e(-\frac{x}{2}) \cdot x^{\frac{\nu}{2}-1}}{2^{\frac{\nu}{2}} \cdot \Gamma(\frac{\nu}{2})}, \quad with \quad x \geq 0 \qquad 6$$

Here $\Gamma(\frac{\nu}{2})$ is the Gamma Function

$$\Gamma(\alpha) = \int_0^{\infty} t^{\alpha-1} \cdot e^{-x}dx, \quad with\, \alpha \geq 0$$

1 The degree of freedom is the number of classes minus 1.

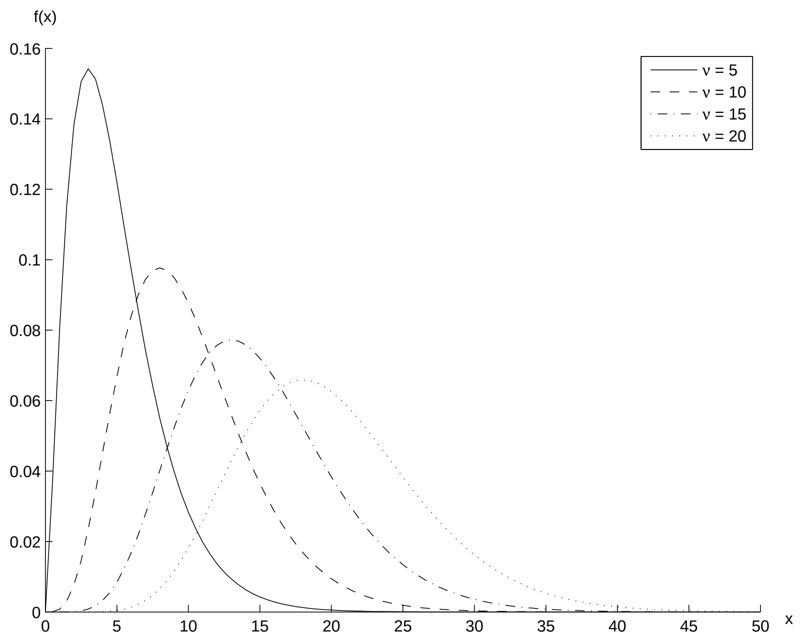


Fig. 2: Chi-square distribution

In the following we will introduce an example to illustrate the Chi-square test.

Example 1:
We toss a coin 150 times and we observe heads 80 times ($O_1 = 80$) and tails 70 times ($O_2 = 70$). The expected value for head and tails according to empirical results is 75. The χ^2 test for the experiment is $\chi^2 = \frac{(80-75)^2}{75} + \frac{(70-75)^2}{75} = 0,66$. Now we want to see whether we can confirm the Zero Hypothesis (the coin is perfect) or not, we assume a level of significance $\alpha = 0.01$. The degree of freedom is 1 since two classes of events are available (head, tails). With the significance level $\alpha = 0,01$ together with the degree of freedom $\nu = 1$ we can read the critical value of the Chi-square test in question. In this case the tabulated critical value for $\chi^2_{0.01}$ is $6,63$. Since $0.66 < 6.63$ we can confirm the Zero Hypothesis.

How the Chi-square test is carried out is summarized in the following steps:

- Carry out the Chi-square test for observation to be tested according to Eq. 5 .
- Determine the degree of freedom.
- Determine the tabulated value (critical value), this critical value depends on two parameters: the degree of freedom ν and the level of significance α.
- Compare the calculated value of the Chi-square test with the critical tabulated value for a given level of significance α. According to the result of the comparison we can either approve or reject the Zero Hypothesis.

A.3 Fisher´s Distribution

The F-distribution was introduced by Sir Ronald A. Fisher and can be used to compare two χ^2 random variables. To do this a third variable being the quotient of the two random variables to be compared with is introduced. If U_1 and U_2 are independent random variables, each having a probability function χ^2 with m and n the degrees of freedom, the random variable

$$D = \frac{\frac{U_1}{m}}{\frac{U_2}{n}} \tag{7}$$

has an F-distribution. The probability density function of the F-distribution is given below

$$f(x) = m^{\frac{m}{2}} n^{\frac{n}{2}} \cdot \frac{\Gamma(\frac{m}{2}+\frac{n}{2})}{\Gamma(\frac{m}{2})\Gamma(\frac{n}{2})} \cdot \frac{x^{\frac{m}{2}-1}}{(mx+n)^{\frac{m+n}{2}}} \tag{8}$$

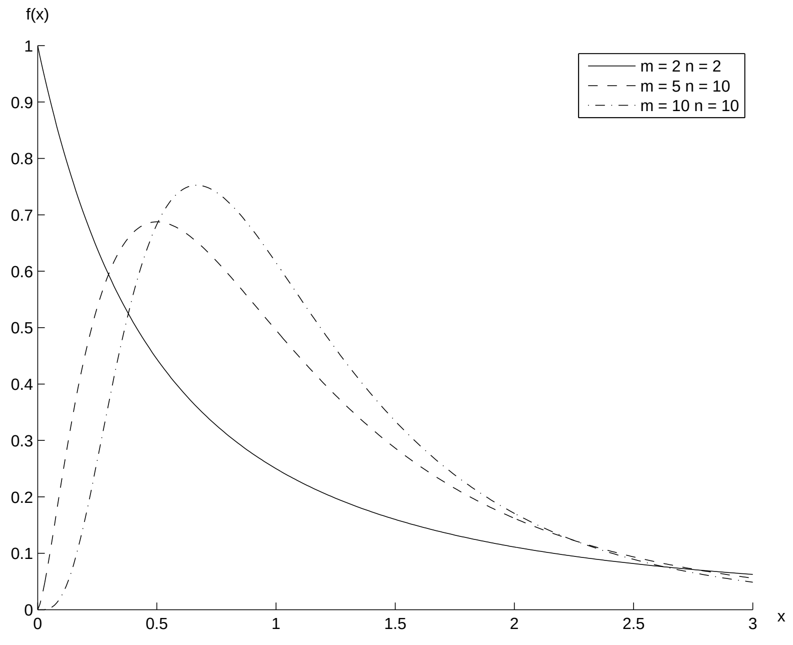
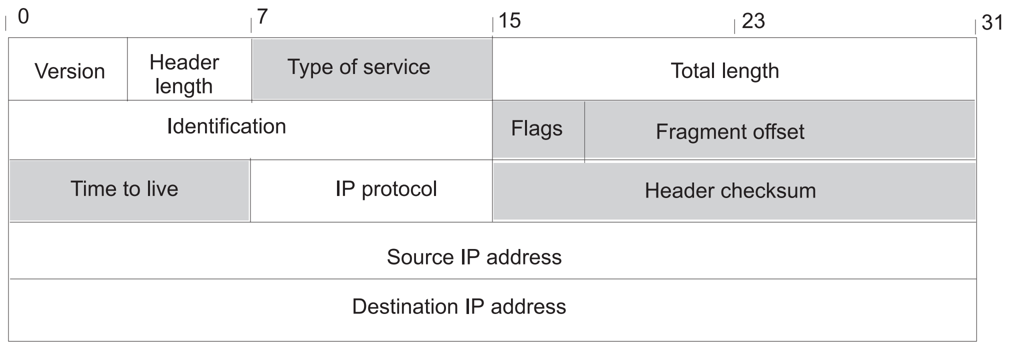


Fig. 3: Fisher´s distribution

Solutions for Exercises

Solution for Exercise 3.2-1:

Unlike to ESP, AH extends its protection to immutable or predictable fields of the outer IP header. It is therefore necessary to zero out the mutable fields prior to computation of the Integrity Check Value. The mutable fields of an IPv4 header which are not included in the computation of the Integrity Check Value are the shaded fields in the figure below. "Mutable" means that the content of this field may be modified during transit from source to destination (e.g. from some router).



The base fields of IPv4 header are:

– Version: immutable (IPv4)

– Internet Header Length: immutable (the IP header length should be constant)

– Type of Service: mutable (Some routers are known to change the value of this field, even thought the IP specification does not consider Type of Service to be a mutable header field.)

– Total Length: immutable

– Identification: immutable (Since this field identifies the data packet in a sequence, it should not be changed.)

– Flags: mutable (This field is excluded of the integrity computation since an intermediate router might set the DF bit, even if the source did not select it.)

– Fragment Offset: mutable (Since AH is applied only to non-fragmented IP packets, the Offset field must be zero. Even thought this field is predictable, it is excluded from the integrity computation.)

- Time to Live: mutable (This field is change on the route as a normal course of processing by routers, and thus its value at the receiver is not predictable by the sender.)
- Protocol: immutable (This should be the value for AH)
- Header Checksum: mutable (This will change if any of these other fields changes, and thus its value upon reception cannot be predicted by the sender.)
- Source Address: immutable
- Destination Adress: Mutable but predictable

Solution for Exercise 3.2-2:

1. IPSec may be implemented in end systems (hosts) or in security gateways such as routers and firewalls.
2. The implementation of IPSec is typically done by directly modifying the IP stack (the communication software). When access to the IP stack of the machine is not possible, IPSec may be implemented as a shim that modifies (protects), and inserts packets to the IP stack. This is called "Bump in the stack" (BITS). IPSec can also be implemented as external, dedicated crypto-device called "Bump in the wire" (BITW).

Solution for Exercise 3.3-1:

1. F-Secure SSH is a commercial version of SSH (Secure Shell) which is widely deployed on the Internet today.
2. Port 22 has been IANA-registered and officialy assigned for SSH.
3. The major components of the SSH (Secure Shell) protocol are the SSH Transport Layer Protocol, the SSH User Authentication Protocol, and the SSH Connection Protocol.
4. The SSH Transport Layer Protocol provides cryptographic host authentication, data confidentiality (encryption) and data integrity. It also supports secure key exchange algorithms.
5. Several data encryption algorithms can be used in SSH (in the SSH Transport Layer Protocol). Mandatory to implement is Triple DES in CBC mode. IDEA in CBC mode, the stream cipher ARCFOUR, and Blowfish in CBC mode are also defined. The algorithm to be used is negotiated during the connection establishment.

6. This prevents attacks. If an attacker eavesdropes the connection and records the traffic, the more data encrypted with one key he has, the more material for crypto-analysis is available to him (the probability for successful crypto-analysis rises).

7. Yes. Different encryption algorithms and keys may be used in each direction of the communication (e.g. algorithm A_1 with key K_1 for the traffic from the server to the client and algorithm A_2 with key K_2 for the traffic from the client to the server). The applied authentication algorithms and keys can also be different for each direction.

Solution for Exercise 3.3-2:

1. A separate port number must be assigned for every application with SSL support. Following dedicated ports have been reserved by IANA:
   - port 443 for HTTP with SSL
   - port 465 for SMTP with SSL
   - port 585 for IMAP with SSL
   - port 636 for LDAP with SSL
   - port 990 for FTP with SSL
   - port 992 for TELNET with SSL
   - port 994 for IRC with SSL
   - port 995 for POP3 with SSL
   and others.
   For more information see http://www.iana.org/assignments/port-numbers

2. SSL consists of four major subprotocols, which operate in two layers. On the higher layer of SSL operate the protocols dealing with SSL management: The SSL Handshake Protocol, the SSL Change Cipher Spec Protocol, and the SSL Alert Protocol. On the lower layer operates the SSL Record Protocol, which in fact provides the security services (encryption and authentication of data).

Solution for Exercise 3.3-3:

1. TLS (Transport Layer Security) is the follow-on Internet standard of SSL, initiated by IETF. It operates in the transport layer and provides cryptographic security, interoperability, extensibility and efficiency.

2. Application protocols secured with TLS use the same ports as if the application was secured with SSL. So, the port numbers are same as those cited in the Solution for Exercise 3.3-2.1.

3. Similar to SSL, TLS is a layerd protocol consisting of four major sub-protocols. On the higher layer of TLS operate the protocols dealing with TLS management: The TLS Handshake Protocol, the TLS Change Cipher Spec Protocol, and the TLS Alert Protocol. On the lower layer operates the TLS Record Protocol.

Solution for Exercise 3.5-1:

1. No. The transport layer security protocols (SSH, SSL, TLS) operate on the top of the transport layer, i.e. between the transport and the application layer. This means that all information (headers) of the underlying protocols (e.g. IP) are unprotected. This enables traffic analysis. For exapmple, by examing an unencrypted IP source and destination addresses and the TCP port numbers, or examining the volume of network traffic flow, a traffic analyst can determine which parties are interacting and what types of services are being used. Since IP headers are transmitted unprotected, IP address spoofing attacks are also possible.

2. SSL and TLS offer security services only for connection-oriented transport, such as those provided by the TCP protocol. Conectionless transport, such as those provided by the UDP protocol, can not be secured with SSL or TLS.

3. **Advantage**: Providing security at the application layer is very flexible, because the scope and the strength of the protection can be tailored to meet the specific needs of the aplication and the user has complete access to the data he wants to protect and to the private keys. To provide the security services, the application can be extended without having to depend on the operating system.

   Disadvantage: The security mechanisms have to be designed independently for each application, i.e. applications have to be enhanced. As each application has to define its own security mechanisms, the probability of making mistakes or not using security at all is greater. (Generally, users prefere convinience and each extra click they have to do leads them very often to refuse using security functions!)

Solution for Exercise 4.4-1:

There are basically three overlapping types of security risks regarding the World Wide Web:

1. Bugs or misconfiguration problems in the Web server that allow unauthorised remote users to:

   - Steal confidential documents.

- Upload files on the server.
- Execute commands on the server host machine, allowing them to modify the system.
- Gain information about the Web server's host machine that will allow them to break into the system.
- Launch denial-of-service attacks, rendering the machine temporarily unusable.

2. Browser-side risks, including:
   - Active content that crashes the browser, damages the user's system, breaches the user's privacy, or merely creates an annoyance.
   - The misuse of personal information knowingly or unknowingly provided by the end-user.
3. Interception of network data sent from browser to server or vice versa via network eavesdropping.

Solution for Exercise 4.4-2:

1. HTTP basic authentication:
   On the first access to an authenticated resource, the server will return a 401 status ("Unauthorized") and include a WWW-Authenticate response header. This will contain the authentication scheme to use and the realm name. The browser would then ask the user to enter a username and a password. It then requests the same resource again, this time including an authorisation header which contains the scheme name ("Basic") and the username and password entered.
   The server checks the username and password, and if they are valid, returns the page. If the password is not valid for that user, or the user is not allowed access because he is not listed on a require user line or in a suitable group, the server returns a 401 status as before. The browser can then ask the user to retry his username and password.
   Assuming the username and password was valid, the user might next request another resource which is protected. In this case, the server would respond with a 401 status, and the browser could send the request again with the user and password details. However this would be slow, so instead the browser sends the Authorization header on subsequent requests. Note that the browser must ensure that it only sends the username and password to further requests on the same server.
   The browser needs to remember the username and password entered, so it can send them with future requests from the same server.

2. HTTP digest authentication:
   Version 1.1 of HTTP (published in July 1996) introduced an improved method called digest authentication. This uses the same exchange of packets as basic authentication, but the "Unauthen- ticated" reply now includes a value, known as the nonce, which acts as a challenge. Instead of replying with the username and password, the client calculates a message digest (using the MD5 algorithm) from the username, password and nonce and returns this along with the username as authentication information. The server then repeats the MD5 calculation, using the user's correct password, and returns the document if the two digests match. To do this the server must hold each user's password in a form suitable for calculating the MD5 digest. It is imperative that these passwords be stored as securely as possible, since anyone possessing them would immediately be able to masquerade as any valid user of the server.

Solution for Exercise 5.3-1:

The notion **Sender anonymity** is provided in the Mix- an in the DC-concept. **Sender anonymity** means that sender A of message m stays anonym for the recipient B as well as for an observing attacker.

The notion **Server anonymity** is provided in the JANUS network. **Server anonymity** enables to hide the identity of a server in a communication session.

The notion **Recipient anonymity** is provided in the Mix- an in the DC-concept. In a communication relationship between two partners A and B **recipient anonymity** allows e.g. A to send a message to B without breaking the anonymity of B. For this purpose A becomes a special address which refers to B but does not reveal $B's$ identity to A.

The notion **Client anonymity** is provided in the JANUS network. **Client anonymity** is reached, when all information about the client is concealed.

Solution for Exercise 6.1-1:

1. Packet filters are not firewall systems in the strong sense, because they do not authorize traffic from inside to outside, and vice versa. Firewall systems should only allow authorized traffic to pass, as defined by the local security policy.

2. A security policy is a document or a statement about security objectives which should be fulfilled in an organization.

   a. A security policy should be explicit and understandable by non-technical persons.

   b. A security policy sets explicit expectations and responsibilities among the staff, the management, users and customers.

   c. A security policy should include consequences that are activated when the policy is not followed.

   d. A security policy needs to describe what you are trying to protect and why. It does not necessarily need to describe technical details of how.

3. A router with packet filter support is called a `screening router`.

4. The following problems arise with IP fragmentation and packet filtering:

   a. Only the first fragment will contain the IP header information, so the packet filter can inspect only the first fragment and can let non-first fragments through.

   b. Another approach is to reassemble the IP packet by the packet filter. In this case denial of service attacks are possible against the packet filter.

   c. An attacker can construct overlapping packets.

Solution for Exercise 7.3-1:

Telnet and FTP do not provide user-to-host authentication. Both Telnet and FTP users must supply login passwords. The passwords are transmitted in cleartext and are therewith subject to eavesdropping. Without the existence of external filtering tools, a Telnet or FTP server initiates connections to both trusted and untrusted networks. Telnet and FTP provide no confidentiality and no data integrity, since the traffic between server and client is not encrypted and can be altered by an intermediate host. If an anonymous FTP server is not properly configured, an unauthorized user can get shell-level access to the server's host. We can avoid all these security risks by the use of the SSH software which provides among other things secure remote terminal access and secure file transfer. Running Telnet over an encrypted network connection such as a VPN connection provides the desired security, too.

Solution for Exercise 7.6-1:

1. Examples of executable content which can affect the network security are binary mail attachments, helper applications and plug-ins, scripting languages, Java applets, and ActiveX.

2. A denial of service attack can be mounted against a host, when a JavaScript code is written in a malicious manner. For example, the JavaScript

code can require too much of the CPU or memory resources, or it can try to open a large number of windows on the victim's screen.

3. Java uses some techniques to limit the risk related to a downloading of Java applets. These techniques are the Java Sandbox, the SecurityManager class, the Bytecode Verifier, and the Java Class Loader.
   - Java Sandbox: enables Java programs to run only on a JVM inside a restricted runtime environment.
   - SecurityManager class: will be called before any unsafe operation is executed.
   - Java Class Loader: examines classes to ascertain that they do not violate the runtime system.
   - Bytecode Verifier: assures whether a downloaded bytecode stems from a valid Java program.

Assignments

Assignments for Chapter "Introduction"

Assignment 1: As mentioned in the course, the significance of security in information technology increases constantly. Find some actual examples from the news about fraudulent activities or abuse in the Internet. **3 P.**

Assignment 2: In the following exercises security goals along with current attack trends are dealt with. **8 P.**

a) Important security goals in open networks are: **4 P.**

- Confidentiality
- Integrity
- Authentication

Can you briefly explain the meaning of these words and give two more goals?

b) Complete the following table by entering which security goals have been violated in the following scenarios. **4 P.**

a. Alice sends an E-mail to Bob. The E-mail is captured on the way from Alice to Bob by Eve. Eve changes the content of the E-mail and sends it to Bob.

b. Alice is talking to Bob on the telephone, the line is eavedroped.

c. Eve gets into Alice´s work place and steals her disc.

d. Alice has a file containing a list of people on her computer that she trusts. They are Bob, Jane, Brad und Paul. Eve intiates a connection to Alice's computer and pretends she is Bob. She then changes the list inscribing her name on it and she then crashes the system.

e. Alice surfs on the internet, her data is forwarded through Eve's Computer (Router). Eve has installed software on her computer that gives her the posssibility to track Alice's surfing behaviour

f. Eve orders a product but claims she did not.

Scenario	Security goals violated
a)	
b)	
c)	
d)	
e)	
f)	

4 P. **Assignment 3**: Every year the Computer Security Institute (CSI) publishes a survey on attacks and their consequences. Briefly comment page 13 (Figure 14) and page 15 (figure 16) of the survey 2005. The survey can be downloaded from http://www.usdoj.gov/criminal/cybercrime/FBI2005.pdf

Assignments for Chapter "Basics"

4+3+2=9 P. **Assignment 4**:

1. What are the main differences between packet switching and circuit switching?
2. In which processes can circuit switching be subdivided?
3. Which packet switching modes are used for the following protocols:
   a. IP
   b. ICMP
   c. UDP
   d. TCP

3+3+3=9 P. **Assignment 5**: We consider the network in Example 2.3-1. Host `A` has IP address `123.76.12.211` and host `B` has IP address `123.76.15.70`.

1. `A` wants to send an IP packet to host `B`. Which source and destination IP addresses have to be specified in the IP packet? Which host's destination MAC address must be used in the packet of the network access layer transporting this IP packet? What is the name of the routing technique used on the network access layer?
2. A server is running on host `B` (port 499) over the TCP protocol. Host `A` wants to open a TCP connection to the server on host `B`. Describe the contents of the TCP packets (port numbers, sequence numbers, TCP flags (SYN, ACK, FIN)) which have to be sent between the two hosts in order to establish the TCP connection. The sequence numbers are starting at 456 (host `A`) and 789 (host `B`), respectively.
3. The connection between `A` and `B` is now established. `A` sends a message with 20 bytes payload to `B` and `B` sends a message with 30 bytes payload to `A` over this connection. Describe the contents of the TCP packets (port numbers, sequence numbers, TCP flags (SYN, ACK, FIN)). No errors occur and all messages have to be acknowledged.

Assignment 6: Consider again the network in Example 2.3-1 and Assignment 5. Host A has IP address `123.76.12.211`, host B has IP address `123.76.15.70` and a TCP service listens on host B on port 499. **2+3+1+3+3=12 P.**

1. How can host A perform a denial of service attack against host B?
2. Describe three ways how this attack can be staffed off!
3. How can host A run an IP spoofing attack against host B?
4. Recite three Internet services which authenticate the clients with unencrypted passwords!
5. What are the standard port numbers of the following Internet services:
   a. SMTP
   b. RAP
   c. FTP

Assignments for Chapter "Internet Security Protocols (Part I)"

Assignment 7: Answer **briefly** the following questions: **12 P.**

1. Which security services does IPSec provide?
2. Where in the TCP/IP protocol stack is the IPSec protection applied?
3. Do the IP packet format change after applying some IPSec security features?
4. Can different traffic flows have different protection level with IPSec?
5. Which encryption algorithms can be used in ESP of IPSec?
6. Which authentication algorithms can be used in ESP of IPSec?
7. Is the SPI field of the ESP header encrypted? Why?
8. Why is the lack of encryption of the Sequence Number field in the ESP header not a security risk?
9. In which mode of ESP it is possible to do traffic analysis of the transmitted packets?
10. What is the difference between the data integrity provided by ESP and AH?
11. Why, from your point of view, no public key authentication algorithms (like RSA or DSS) have been defined for use with AH?

12. Does the key management system (IKE) is dependent on the applied authentication and encryption mechanisms?

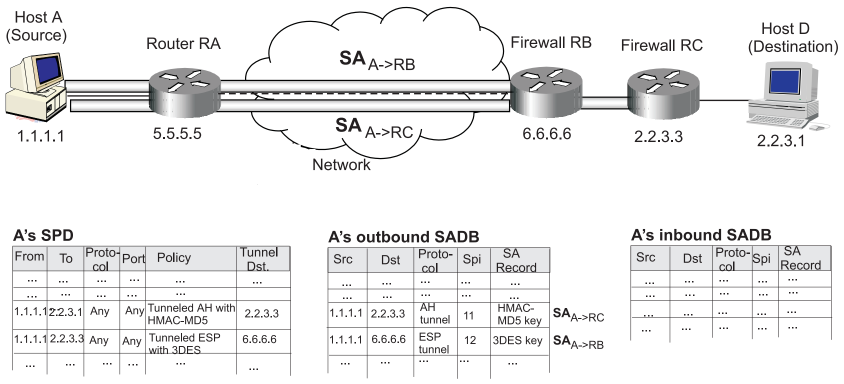
2 P. **Assignment 8**:

1. What is Security Association (SA) bundle? How can Security As sociations be combined in bundles?
2. The IPSec architecture document states that when two transport mode SAs are bundled to allow both AH and ESP protocols on the same end-to-end flow, only one ordering of security protocols seems appropriate: performing the ESP protocol before performing the AH protocol. Why is this approach recommended rather than authentication before encryption?

3+3=6 P. **Assignment 9**: Draw a flow diagram with the most important steps of the

1. outbound IPSec processing of IP packets,
2. inbound IPSec processing of IP packets.

5 P. **Assignment 10**: Consider the scenario showed in the following figure. Suppose that host A sends a packet to destination D.



A's SPD

From	To	Proto-col	Port	Policy	Tunnel Dst.
...	...	...	...	...	...
...	...	...	...	...	
1.1.1.1	2.2.3.1	Any	Any	Tunneled AH with HMAC-MD5	2.2.3.3
1.1.1.1	2.2.3.3	Any	Any	Tunneled ESP with 3DES	6.6.6.6
...	...	...	...	...	...

A's outbound SADB

Src	Dst	Proto-col	Spi	SA Record	
...	...	...	...	...	
...	...	...	...	...	
1.1.1.1	2.2.3.3	AH tunnel	11	HMAC-MD5 key	**SA**$_{A\text{->}RC}$
1.1.1.1	6.6.6.6	ESP tunnel	12	3DES key	**SA**$_{A\text{->}RB}$
...	...	...	...	...	

A's inbound SADB

Src	Dst	Proto-col	Spi	SA Record
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

1. Which IPSec processing has to be done on the outbound packet **in the host** A according to the entry in its Security Policy Database?
2. Sketch the order of headers of the outbound packet wich leave host A!

Assignments for Chapter "Internet Security Protocols (Part II)"

Assignment 11: Describe **briefly** how SSH authentication protocol runs! **2 P.**

Assignment 12: **2 P.**

1. Where in the TCP/IP protocol stack does the SSL protocol run?
2. Which control data (headers) are protected with SSL, beside the application data?

Assignment 13: **3 P.**

1. Explain **briefly** the steps of the SSL record protocol!
2. What kind of message can be the encrypted message of the SSL record?

Assignment 14: Explain **briefly** the four phases of the SSL Handshake Protocol! **8 P.**

Assignment 15: *Additional exercise* **0 P.**

There are many sites in Internet where personal and sensitive user data is required. Examples are online-shops or insurance companies ("give me your data and I'll find the best offer for you"). Examples are www.amazon.de, www.lob.de, www.versicherungsvergleich.de, and others. Some of them use the SSL protocol for protecting the data in transit (e.g. www.amazon.de) but the majority does not do this yet. Find some examples where the SSL protocol is used to protect the data and some examples where the data is not protected at all! Be aware when sending your personal data and do this only if data protection is guaranteed!

Assignments for Chapter "World Wide Web Security"

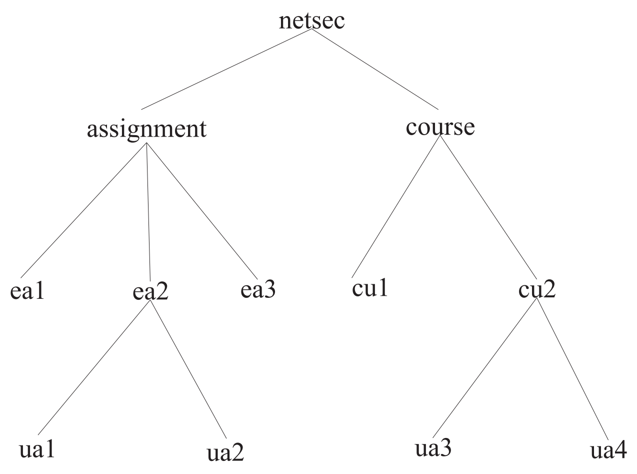
Assignment 16: Using the program `htpasswd` we have created a file containing the following usernames and passwords (`/etc/apache/passwd`): **5+5=10 P.**

```
alice:cXU3jeeggMprU
bob:tX1WkwIxq.efg
charly:dYZQ3GIFAaQfA
cryptoad:B0ENPe.o4SLdg
```

Furthermore, there exists a group file (`/etc/apache/htgroup`) containing the three groups `all`, `students1`, and `students2`:

```
all: alice bob charly cryptoad
students1: alice bob charly
students2: alice bob
```

The directory structure looks like this:



What are the contents of the `.htaccess` files and where to put them to restrict user access in the following cases?

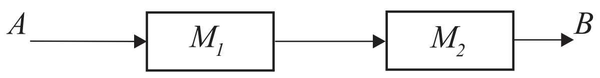
1. Case 1
   - The administrator `cryptoad` has access to all directories.
   - No one but the administrator `cryptoad` has access to the directory `assignement` and all subjacent directories.
   - `alice` and `bob` have access to the directories `course`, `cu2`, `ua3`, and `ua4`.
   - `charly` has access to the directories `course` and `cu1`.
   - The directory `netsec` is not access restricted.
2. Case 2
   - The administrator `cryptoad` has access to all directories.
   - `alice`, `bob`, and `charly` have access to the directories `assignment`, `ea2`, `ua1`, and `ua2`.
   - The following directories are not access restricted: `netsec`, `course`, and all directories below `course`.

4 P. **Assignment 17**: How does cookie data move?

Assignment 18: How do web sites use cookies? **5 P.**

Assignments for Chapter "Anonymity Techniques"

Assignment 19: We assume that one message can be sent anonymously using the mix concept. We want to send anonymously a message $m = 124$ using a sequence of 2 mixes as illustrated below: **4 P.**



Our mix concept is based on RSA public key cryptography. The public key $k_{e,A}$, $k_{e,B}$, k_{e,M_1}, and k_{e,M_2} respectively is the pair $k_{e,A} = (e_A, n_A) = (5, 62894113)$, $k_{e,B} = (e_B, n_B) = (5, 28829)$, $k_{e,M_1} = (e_{M_1}, n_{M_1}) = (5, 55465219)$, and $k_{e,M_2} = (e_{M_2}, n_{M_2}) = (3674911, 6012707)$ respectively. The private keys are $k_{d,A}$, $k_{d,B}$, k_{d,M_1}, and k_{d,M_2} are respectively $k_{d,A} = d_A = 37726937$, $k_{d,B} = d_B = 22781$, $k_{d,M_1} = d_{M_1} = 44360237$, and $k_{d,M_2} = d_{M_2} = 422191$ respectively.

a) Which condition must the two mixes satisfy in order to reach an anonymous sending of a message *m*. **1 P.**

b) Suppose that at each encryption the message to be sent need not to be attached with any random bits and that the addresses are encrypted independently of the message. Compute the incoming and the outcoming message (without address) of the mix M_1. **3 P.**

Assignment 20: Which problems can occur when the basic idea of the DC concept is implemented in a network? **2 P.**

Assignments for Chapter "Packet filters and firewall systems"

1+3+3=7 P. **Assignment 21**:

1. What is stateful packet filtering?
2. What problems remain and arise with stateful packet filtering?
3. What are the security related advantages of NAT (Network Address Translation)?

14 P. **Assignment 22**: Recall Example 6.1-2. Suppose our intranet has the subnet IP number `172.16.1.0` with subnet mask `255.255.255.0`. The users in the local subnet are not allowed to access any services on the Internet. The users on the Internet are allowed to access the local WWW server on the machine with IP number `172.16.1.10`.

Write down the commands to build a packet filter with `iptables` on a router or a dual-homed host between the Internet and the local subnet!

Assignments for Chapter "Application Layer Security (Chapter 7.1 - 7.6)"

2 P. **Assignment 23**: Which layer of the Internet model is well suited for providing the security of the Internet services and applications? Justify briefly your answer through some examples!

2+2+2=6 P. **Assignment 24**:

1. How does a distributed file system with NFS work?
2. Give some security weaknesses of NFS Version2!
3. Give two secure distributed file systems and explain how they work!

1+1+2=4 P. **Assignment 25**:

1. What are the cryptographic security services provided by S/MIME for electronic messaging applications?
2. What is the first step to do before signing or enveloping a MIME entity?
3. S/MIME defines two formats for signing messages. What is the main difference between these two formats?

Assignment 26: **1+1+2=4 P.**

1. Give examples of executable content which can affect the network security!
2. How can JavaScript be used to launch a denial of service attack against a host?
3. Describe shortly how can the risk related with download of Java applets be limited!

Assignments for Chapters "Application Layer Security and Security in Wireless and Mobile Networks (Chapter 7.7 - 8.2)"

Assignment 27: Which security services are provided by GSM? Give a brief description how these services are realized in GSM. **3 P.**

Assignment 28: Answer the following questions concerning GSM authentication: **15 P.**

1. Which entities are involved in GSM authentication?
2. Give an overview of the authentication protocol that is used in GSM.
3. Which parties are authenticated and how?

Assignment 29: Make an online search for attacks on GSM Systems. Give a brief description for at least three of these attacks and associate them with one ore more of the following security goals: **12 P.**

- Integrity
- Confidentiality
- Authenticity
- Authorisation
- Nonrepudiation
- Availability
- Anonymity

Example:
The tap-structure attacks [Shamir99] mentioned in the course can be briefly described as real time cryptoanalysis attacks on the strong version of the `A5` algorithm, the `A5/1` variant. These two attacks are based on subtle flaws in the tap structure of the linear feedback shift registers. In the first variant the key can be computed from two minutes of captured conversation in about one second on a single PC. The second variant needs the captured data of two seconds of conversation and computes the key within several minutes. In both variants the

computation has to be carried out only once. Thenceforward the communication can be eavesdropped in real time.

This attack compromises the security goal of confidentiality.

Remember to mention the references for your examples.

Assignments for Chapter "Security in Wireless and Mobile Networks (Chapter 8.3 - 8.4)"

6 P. **Assignment 30**: What are the fundamental improvements in security features of UMTS compared to GSM?

6 P. **Assignment 31**: UMTS provides mutual authentication between the USIM and the network. Give a brief answer to the following questions concerning authentication in UMTS:

2 P. a) Which attacks (known from GSM) does the introduction of mutual authentication prevent?

4 P. b) Have you an idea why this kind of attacks are still possible during the transition phase from 2G (GSM) to 3G (UMTS) technology? Make an online search.

8 P. **Assignment 32**: Make an online search for attacks on WAP Systems. Give a brief description of at least two of these attacks and associate them with the following security goals of WAP:

- Data Integrity
- Confidentiality
- Authenticity
- Nonrepudiation

Assignments for Chapter "Security in Wireless and Mobile Networks (Chapter 8.5 - 8.6)"

2 P. **Assignment 33**: A common attack against a cipher suite is a known plaintext attack. What is in general a known plaintext attack? Is it possible to find a plaintext associated with it's ciphertext in a wireless network secured with WEP?

Assignment 34: Outline the structure of the decryption engine of WEP. **4 P.**

Assignment 35: Make an online search for attacks on WAP Systems. Give a brief description of at least two of these attacks and associate them with the following security goals of WAP: **8 P.**

- Data Integrity
- Confidentiality
- Authenticity
- Nonrepudiation

Assignments for Chapter "Electronic Payment Systems"

Assignment 36: What is an electronic payment system? List the main categories of electronic payment systems and explain why Credit Card-based systems need much stronger security than other electronic payment systems. **10 P.**

Assignment 37: **4*5=20 P.**

1. What is the function of the *ODsalt* contained in *OIData* in the *SET* protocol?
2. What is the main feature of electronic check systems? Is the NetBill server a bottleneck of the hole system and why?
3. List the main parties and main protocols in an electronic cash payment system. What is the function of the *Payer_code* in Ecash systems?
4. What is the main idea behind electronic micropayment systems? How do the electronic micropayment systems differ from macropayment systems introduced earlier in this chapter? What is the data field *ID# in the Millicent script used for?*

Assignments for Chapters "Security Aspects in Mobile Agents Systems and Copyright Protection"

Assignment 38: What is a mobile agent? **2 P.**

Assignment 39: Explain how a malicious host can attack a mobile agent? **3 P.**

2 P. **Assignment 40**: What are the research areas of the mobile agent paradigm?

3 P. **Assignment 41**: What kind of communication paradigm can be applied to a mobile agent system when one wants to communicate with a mobile agent and how should the proposed communication paradigm be implemented (i.e. Please explain only some main requirements of your solution)?

Assignments for Chapter "Intrusion Detection"

10 P. **Assignment 42**: The purpose of this exercise is to make you familiar with tools used to detect intrusions.

3 P. a) Download Nmap from http://www.insecure.org/nmap/download.html and perform a scan on the machine `bitburger.fernuni-hagen.de`.

1. Summarize the scan result.
2. Which ports does Nmap reports open? Which of them is useless in your opinion.
3. Which operating system is in use?

7 P. b) Ethereal is one of the most powerful tools used to sniff packets traversing a network for maintenance and troubleshooting purposes. Download Ethereal from http://www.ethereal.com.

Start Ethereal, open the file `test.cap` and analyse the trace of the capture by answering the following questions.

1. Which protocols can you identitfy in the trace?
2. We consider packet 30. What are the hexadecimal values of the source and destination address (Ethernet address)?
3. Determine the fields of the 30th packet according to Fig. 12.2-2 and Fig. 12.2-4. Has this packet been fragmented?
4. The packet 30 has a captured length of 810 bytes, how do we get this?
5. Filter the packets by entering `http`. Which methods can you identify in the trace? Analyse the packets 93 and 95, what is the Unified Resource Identifier (URI)?
6. Determine the HTTP version the server is running, which status code was returned from the server to your browser.
7. Determine the following statistics: average packet size, average byte/s, total amount of TCP and UDP packets captured.

Assignment 43: In anomaly detection the normal behaviour of a variable is modeled. Generally to do this, many attributes which characterise the variable are taken into consideration. Possible attributes for network based anomaly detection are interarrival time of packets, sender address, receiver address, number of bytes, number of packets, etc. For reasons of simplicity we will only consider two attributes in this exercise. **15 P.**

A web server has been observed for a while and the following values have been recorded for different time intervals of equal length δt_i (see Table 1). Answer the following questions.

Tab. 1: *Profile of the web server*

Zeit	Number of packets received	Number of bytes received (Bytes)
δt_0	13	10498
δt_1	12	10478
δt_2	11	10480
δt_3	13	10476
δt_4	12	10454
δt_5	12	10472
δt_6	12	10486
δt_7	13	10483
δt_8	11	10492
δt_9	12	10467
δt_{10}	13	10484
δt_{11}	13	10475
δt_{12}	13	10476
δt_{13}	13	10465
δt_{14}	12	10460
δt_{15}	11	10469
δt_{16}	12	10473
δt_{17}	11	10472
δt_{18}	10	10472
δt_{19}	13	10471

a) Complete the frequency tables for both the received number of packets as well as received number of bytes. **2 P.**

Tab. 2: *Frequency table for number of bytes received*

	Frequency	Percentage
10454	1	5,0
10460	1	5,0

Tab. 3: *Frequency table for number of packets received*

	Frequency	Percent
10	1	5,0

3 P. b) Draw a histogram for the attribute number of bytes received.

3 P. c) Approximate the histogram through a normal distribution and determine the confidence interval for a level of significance $\alpha = 0.05$. Can this normal distribution be used for anomaly detection?

Solution hints: The histogram and the normal distribution can be drawn using the Statistical Software Packet SPSS (Statistical Product and Service Solution) which is freely available for students of the FernUniversity in Hagen (http://www.fernuni-hagen.de/urz/service/Liste.html).

1 P. d) In the following a Multivariate Analysis using the Hotelling's test is taken into consideration. Give the Matrix of observations M according to equation Eq. 12.4-4 based on the samples in Table 1 .

e) Calculate the first factor in equation Eq. 12.4-5 **2 P.**

$$(X_1 - \bar{X})(X_1 - \bar{X})^t$$

f) The variance-covariance matrix equals **4 P.**

$$S = \begin{bmatrix} 0,83 & 1,09 \\ 1.09 & 106.98 \end{bmatrix}$$

Apply the Hotelling's test to the observations x_{20} at time interval Δt_{20} and x_{21} at time interval Δt_{21} in Table 4 for a level of significance $\alpha = 0.05$. Are these observations anomalies?

Tab. 4: *Obervations to be tested*

	Number of packets received	Number of bytes received
Δt_{20}	13	10498
Δt_{21}	12	10478

Assignment 44: Visit the Snort homepage http://www.snort.org/, install Snort and try to be familiar with it. Download the Snort user manual and read it carefully. **5 P.**

a) Interpret the following Snort rules taken from the `snort.conf` file and determine the priority class for each time. **5 P.**

1.
```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-ATTACKS /usr/bin/id command attempt";
flow:to_server,established;
content:"/usr/bin/id"; nocase;
classtype:web-application-attack; sid:1332; rev:5;)
```

2.
```
alert tcp $EXTERNAL_NET any -> $HOME_NET 22
(msg:"EXPLOIT ssh CRC32 overflow NOOP";
flow:to_server,established;
content:"|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|";
reference:bugtraq,2347; reference:cve,2001-0144;
reference:cve,2001-0572; classtype:shellcode-detect;
sid:1326; rev:6;)
```

3.
```
alert tcp $SQL_SERVERS 3306 -> $EXTERNAL_NET any
(msg:"MYSQL server greeting"; flow:from_server,established;
content:"|00|"; depth:1; offset:3;
flowbits:set,mysql.server_greeting; flowbits:noalert;
reference:bugtraq,10655;
reference:url,www.nextgenss.com/advisories/mysql-authbypass.txt;
classtype:attempted-user; sid:3665; rev:1;)
```

4.
   ```
   alert ip $EXTERNAL_NET any -> $HOME_NET any
   (msg:"BAD-TRAFFIC Unassigned/Reserved IP protocol";
   ip_proto:>134;
   reference:url,www.iana.org/assignments/protocol-numbers;
   classtype:non-standard-protocol; sid:1627; rev:3;)
   ```

5.
   ```
   alert tcp $EXTERNAL_NET any -> $HOME_NET 79
   (msg:"BACKDOOR CDK"; flow:to_server,established;
   content:"ypi0ca"; depth:15; nocase; reference:arachnids,263;
   classtype:misc-activity; sid:185; rev:5;)
   ```

4 P. **Assignment 45**: Write a Snort rule to detect the following attacks.

3 P. a)

1. A packet sent to port 80 with the content „cmd.exe“ is in principle a good sign for an attack on a windows based web server e.g. Internet Information Server (IIS). With „cmd.exe“ a Windows-Shell can be invoked. Write a rule to identify a directory traversal as seen in Example 12.3-1 and Example 12.3-2.

2. The Code Red worm has been dealt with in Example 12.2-5. The worm has the following signature. Write down a rule to indentify the Code Red worm.

   ```
   default.ida?NNNNNNNNNNNNNNNNNNNNNNNNN
   NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
   NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
   NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
   NNNNNNNNNNNNNNNNNNNNNNNNNNN
   %u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3% u7801%u9090%
   u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u531
   b%u53ff%u0078%u0000%u00=a
   ```

3. Somebody is scanning your network, write a rule to identify a SYN/FIN scan.

4. As we have seen in Section 12.2.2.1 if an attacker came into possession of the shadow password file `/etc/shadow`, he/she can carry out a dictionary attack to find valid passwords. Write a Snort rule to detect attempts to access the `/etc/shadow` file.

Solutions for Assignments

Solution for Assignment 1:

Look at:

- http://www.heise.de/security/
- http://www.securityfocus.com/
- http://www.microsoft.com/germany/technet/sicherheit/bulletins/default.mspx

Solution for Assignment 2:

a) Explanations for the security goals are:

- Confidentiality: The only person allowed to see the information is the person for whom the information is determined.
- Integrity: modifications can be recognised.
- Authentication: identity check.

Further security goals are: availability, anonymity and nonrepudiation.

- Availability: The availability of resources or services is of great importance nowadays e.g. with regard to web presence.
- Anonymity: Keep the identity of the user secret.
- Nonrepudiation: The sender cannot deny having sent the message and the recipient cannot deny having received it.

b) Security goals violated in the scenario above are outlined in the table below:

Scenario	Security goals violated
a)	Integrity
b)	Confidentiality
c)	Availability
d)	Authentication, Integrity, availability
e)	Anonymity
f)	Nonrepudiation

Solution for Assignment 3:

As can be seen from figure 14 (page 13) the frequency of reported attacks has steadily decreased from 1999 to 2005 apart from the misuse of the Wireless Network. The trend with regard to website defacement is also alarming. Although the frequency of reported Malware (viruses, worms, trojans) has decreased as well, the level is still very high and the costs related to Malware are approximately 33% of all costs. Moreover Malware, unauthorized access and theft of proprietary information makes up nearly 90% of all costs.

Solution for Assignment 4:

1. The main differences between packet switching and circuit switching are:

   a. Circuit switching:
   In circuit switching, the communication processes `A` and `B` exclusively use a communication link or channel across the network for the whole communication process. No other process can use this link or channel at the same time.

   b. Packet switching:
   In packet switching, the message which is to be transmitted from `A` to `B` is divided into packets. Each packet is provided with destination and control information and is separately transmitted via the network. Packet switching can be realized in two different operational modes: connectionless and connection-oriented.

2. Circuit switching can be subdivided into the following three processes:

   a. Setup phase: `A` indicates to the concerned switching unit that he wants to communicate with `B`. The switching unit informs `B` about `A`'s request. If `B` negotiates, the link between `A` and `B` is exclusively reserved.

   b. Connection phase: `A` and `B` exchange information over the channel.

   c. Termination phase: Both communication partners can inform the switching unit that the communication is finished.

3. Packet switching modes:

   a. IP: connectionless

   b. ICMP: connectionless

   c. UDP: connectionless

   d. TCP: connection-oriented

Solution for Assignment 5:

We consider the network in Example 2.3-1.

1. Hosts `A` and `B` are on different subnetworks. Nevertheless, the source IP address of the IP packet is `123.76.12.211` and the destination IP address is `123.76.15.70`. The destination MAC address is the MAC address of the interface (binded with IP address `123.76.12.254`) of the router. The routing technique is known as `indirect routing`.

2. The following TCP packets are sent between A and B:

   a. `A` to `B`: (source port=1250, destination port=499, sequence number=456, acknowledgement number=random, SYN=1, ACK=0, FIN=0)
   The source port is a random number between 1024 and 5000.

   b. B to A: (source port=499, destination port=1250, sequence number=789, acknowledgement number=457, SYN=1, ACK=1, FIN=0)

   c. A to B: (source port=1250, destination port=499, sequence number=457, acknowledgement number=790, SYN=0, ACK=1, FIN=0)

3. The following TCP packets are sent between A and B:

   a. A to B: (source port=1250, destination port=499, sequence number=457, acknowledgement number=790, SYN=0, ACK=1, FIN=0)

   b. B to A: (source port=499, destination port=1250, sequence number=790, acknowledgement number=477, SYN=0, ACK=1, FIN=0)

   c. A to B: (source port=1250, destination port=499, sequence number=477, acknowledgement number=820, SYN=0, ACK=1, FIN=0)

Solution for Assignment 6:

Consider again the network in Example 2.3-1 and Assignment 5. Host A has IP address `123.76.12.211`, host B has IP address `123.76.15.70` and a TCP service listens on host B on port 499.

1. A denial of service attack by A against B: Host A continuously sends TCP packets with destination IP address `123.76.15.70` and destination port 499.

2. Possibilities to stave the attack:

   a. Disconnect host B from the network.

   b. Disable the TCP service on port 499 on host B.

   c. Allowing a maximum number of connections on port 499.

   d. Disconnected the subnetwork `123.176.15` is from the router.

   e. May be the best choice: Configure the router with a firewall (packet filter), so that only certain IP, TCP or UDP packets can get into the subnetwork `132.176.15`.

3. Host A sends TCP packets to host B with a source IP address which is not equal to its own IP address `123.76.12.211` and destination port 499. Host A selects an destionation IP address in the packets which is not blocked by the router.

4. Examples of Internet services which authenticate the clients with unencrypted passwords:

   a. Remote Login: Telnet, rlogin, . . .

   b. File Transfer: FTP (File Transfer Protocol)

   c. Mail Message Download: POP (Post Office Protocol)

5. In [RP94] the port numbers of the different Internet services are standardized as follows:
   a. SMTP (Simple Mail Transfer Protcol): port 25 for TCP and UDP.
   b. RAP (Route Access Protocol): port 38 for TCP and UDP.
   c. FTP (File Transfer Protocol) has two ports (channels): a control and data transfer channel:
      i. FTP (File Transfer for Control): port 21 for TCP and UDP.
      ii. FTP-DATA (File Transfer for Data): port 20 for TCP and UDP.

Solution for Assignment 7:

1. IPSec provides the following security services:
   - Data source authentication,
   - Data integrity,
   - Confidentiality (encryption),
   - Protection against replay attacks, and
   - Secure exchange of keys.
2. IPSec protection takes place at layer 2 (IP layer). More exactly it is applied in the TCP/IP stack below the transport layer processing (and with it below the application data) and above the IP routing processing.
3. Yes. The security features of IPSec are implemented as extension headers (AH or ESP) that follow the main IP header.
4. Yes. The granularity in the IPSec policy, maintained by the Security Policy Database (SPD), allows different traffic flows to have different protection levels.
5. Different encryption algorithms can be used. Examples are DES, triple DES, Blowfish, RC5, IDEA, triple IDEA, CAST and others (all in CBC mode). Mandatory to implement is DES in CBC mode.
6. Different authentication algorithms can be used. Mandatory to implement are HMAC-MD5 and HMAC-SHA.
7. The SPI field of ESP header is authenticated but not encrypted. This is a necessity because the SPI is used to identify the state (encryption algorithm and key) used on the other (recepient) side to decrypt the packet.

8. The sequence number provides anti-reply services to ESP, i.e. we want to determine whether a packet is a duplicate. The sequence number does not really require confidentiality because it does not expose any secrets in clear-text form. A sefety check can be accomplished by doing the anti-reply check prior to decryption. Hence, the packet will be droped (if necessary) from the destination without expending resources to decrypt it.

9. When ESP is applied in transport mode it is possible to do trafffic analysis on the transmitted (and encrypted) packets. This is because in this mode the IP header, which contents all traffic information (source address, destination address, protocol), is not encrypted.

10. The AH authenticates portions of the outer IP header, which are not authenticated when using ESP (Compare to Fig. 3.3-6 and Fig. 3.3-8)

11. This is due to the cost: Public-key algorithms are too slow for bulk data authentication.

12. No. The authentication and encryption algorithms to be employed have been specified independent of any specific key management system. The key management mechanism is coupled to the authentication and encryption algorithms only through the SPI field in the ESP or AH header.

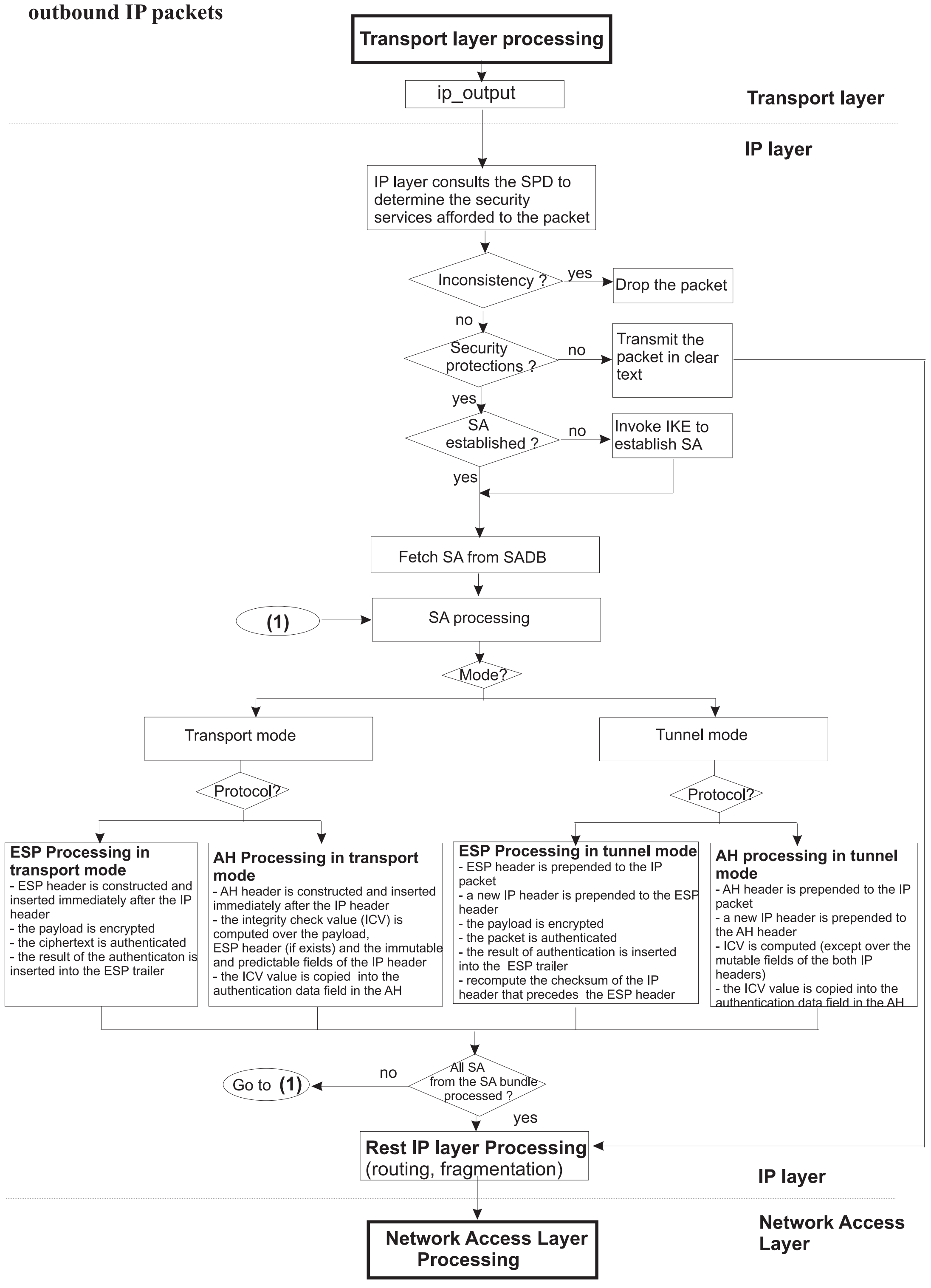
Solution for Assignment 8:

1. Sometimes, several Security Associations (SA) must be employed to the same traffic flow in order to achieve the desired IPSec services. This set of Security Associations (the several steps of IPSec processing) is called Security Associations bundle. SAs can be combined in bundles in two ways: to achieve transport adjacency (applying of more than one security protocol to the same IP packet, without tunelling) or to achieve iterated (multiple) tunelling(applying more than one security protocol to the same IP packet with tunneling).

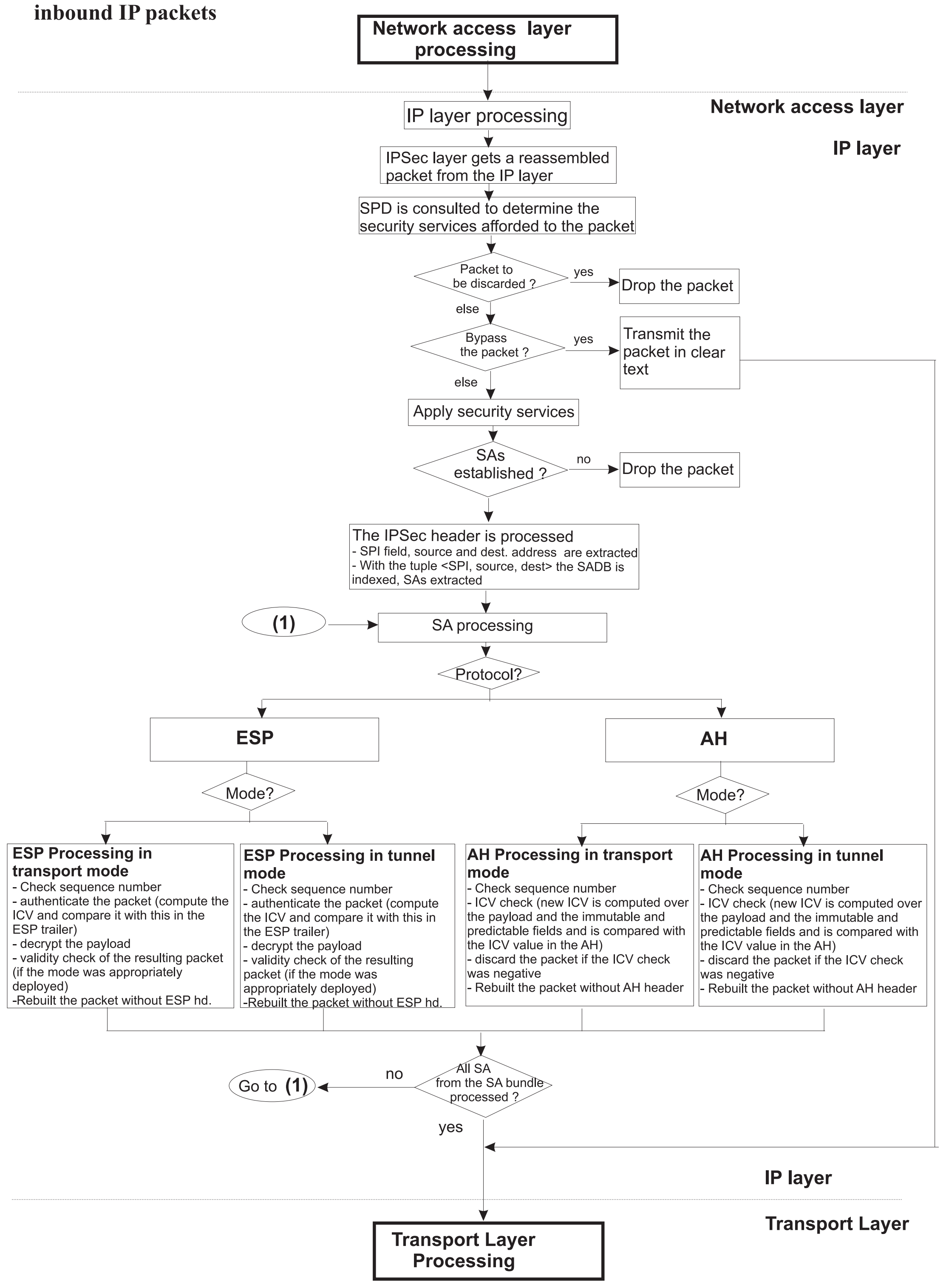
2. The AH header always protects ESP header, not the other way around. The authentication (AH protocol) is a computationaly "cheaper" operation than encryption (ESP protocol). So, first the authentication check is done, and if it fails, the packet will be dropped and no decryption will be performed.

Solution for Assignment 9:

IPSec processing of outbound IP packets


Transport layer processing
ip_output
Transport layer
IP layer
IP layer consults the SPD to determine the security services afforded to the packet
Inconsistency ?
yes
Drop the packet
no
Security protections ?
no
Transmit the packet in clear text
yes
SA established ?
no
Invoke IKE to establish SA
yes
Fetch SA from SADB
(1)
SA processing
Mode?
Transport mode
Tunnel mode
Protocol?
Protocol?
ESP Processing in transport mode
- ESP header is constructed and inserted immediately after the IP header
- the payload is encrypted
- the ciphertext is authenticated
- the result of the authenticaton is inserted into the ESP trailer
AH Processing in transport mode
- AH header is constructed and inserted immediately after the IP header
- the integrity check value (ICV) is computed over the payload, ESP header (if exists) and the immutable and predictable fields of the IP header
- the ICV value is copied into the authentication data field in the AH
ESP Processing in tunnel mode
- ESP header is prepended to the IP packet
- a new IP header is prepended to the ESP header
- the payload is encrypted
- the packet is authenticated
- the result of authentication is inserted into the ESP trailer
- recompute the checksum of the IP header that precedes the ESP header
AH processing in tunnel mode
- AH header is prepended to the IP packet
- a new IP header is prepended to the AH header
- ICV is computed (except over the mutable fields of the both IP headers)
- the ICV value is copied into the authentication data field in the AH
All SA from the SA bundle processed ?
no
Go to (1)
yes
Rest IP layer Processing (routing, fragmentation)
IP layer
Network Access Layer
Network Access Layer Processing

IPSec processing of inbound IP packets

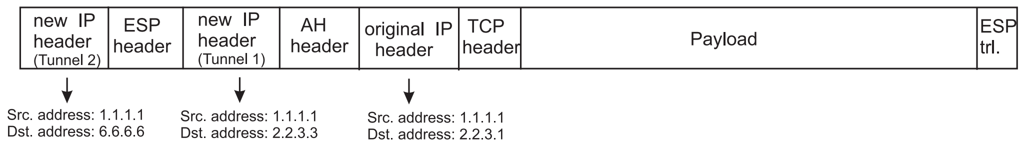

Network access layer processing
Network access layer
IP layer
IP layer processing
IPSec layer gets a reassembled packet from the IP layer
SPD is consulted to determine the security services afforded to the packet
Packet to be discarded ?
yes
Drop the packet
else
Bypass the packet ?
yes
Transmit the packet in clear text
else
Apply security services
SAs established ?
no
Drop the packet
The IPSec header is processed
- SPI field, source and dest. address are extracted
- With the tuple <SPI, source, dest> the SADB is indexed, SAs extracted
(1)
SA processing
Protocol?
ESP
AH
Mode?
Mode?
ESP Processing in transport mode
- Check sequence number
- authenticate the packet (compute the ICV and compare it with this in the ESP trailer)
- decrypt the payload
- validity check of the resulting packet (if the mode was appropriately deployed)
-Rebuilt the packet without ESP hd.
ESP Processing in tunnel mode
- Check sequence number
- authenticate the packet (compute the ICV and compare it with this in the ESP trailer)
- decrypt the payload
- validity check of the resulting packet (if the mode was appropriately deployed)
-Rebuilt the packet without ESP hd.
AH Processing in transport mode
- Check sequence number
- ICV check (new ICV is computed over the payload and the immutable and predictable fields and is compared with the ICV value in the AH)
- discard the packet if the ICV check was negative
- Rebuilt the packet without AH header
AH Processing in tunnel mode
- Check sequence number
- ICV check (new ICV is computed over the payload and the immutable and predictable fields and is compared with the ICV value in the AH)
- discard the packet if the ICV check was negative
- Rebuilt the packet without AH header
Go to (1)
no
All SA from the SA bundle processed ?
yes
IP layer
Transport Layer
Transport Layer Processing

Solution for Assignment 10:

1. In this scenario, the policy indicates that for the host A to send packet to the host D, it has to first authenticate to the router RC and send the packet encrypted to firewall RB. First, the IPSec layer builds a tunneled AH packet to the router RC. After constructing the AH header, the IP layer is invoked to add

the tunnel header. It uses the router RC address (`2.2.3.3`) as the destination address in the selector field. The policy indicates that the packet has to be encrypted and tunneled to RB (`6.6.6.6`). IPSec processes the packet destined to `2.2.3.3`, adding the ESP header, ESP trailer and encryting the payload. It then invokes the IP layer to add an additional header. It uses the router RB address as the destination. This packet with multiple tunnels is then dispatched.

2.



Solution for Assignment 11:

The SSH authentication protocol runs on the top of the SSH transport layer protocol, which has already authenticated the server machine, established an encrypted communication channel, and computed a unique session identifier for this session. When a client declares its user name and the service it wants to use, the server requires user authentication. The server drives the authentication by telling the client which authentication methods can be used (password authentication, SecurID authentication, S/Key and OPIE one-time password authentication, authentication with public key, host-based or Kerberos authentication). The client has the freedom to try the methods listed by the server in any order. The server should have a timeout for authentication, and disconnect if the authentication can not be accepted within this period. Additionaly, the implementation should limit the number of failed authentication attempts. The dialogue continues untill access has been granted or denied.

Solution for Assignment 12:

1. The SSL protocol operates on top of the transport layer, i.e. between the transport and the application layer.

2. The SSL protocol protects the application data and the control data (header) from the application layer. The control data of all protocolls operating on the lower layers (transport layer, IP layer, network access layer) can not be protected with SSL.

Solution for Assignment 13:

1. After the SSL handshake protocol has been run and the both sides share secret keys for encryption and authentication, the **SSL record protocol** starts. It receives data from the higher layers in blocks of arbitrary size. The first purpose of the Record Protocol is **to fragment** this data into SSL plaintext records of 2^{14} bytes or less. Then all records are compressed using the compression algorithm defined in the current session state. Then a **MAC is computed** over the compressed data. After the MAC computation, the compressed message plus the MAC are **encrypted** using symmetric encryption. Just as in the case of MAC, the encryption algorithm to be used is defined in the current CipherSpec and the shared secret key has been agreed in the Handshake Protocol. The final step of SSL Record Protocol processing is to **prepend an SSL record header**, which contains important control information about the protocol. This structure of authenticated and encrypted fragment with SSL header is known as **SSL record**. This record is now proceeded to the lower layer protocol.

   At the receiving site, the operations in the SSL Record Layer are performed vice versa: When the SSL record is received, first the SSL record layer header is removed, then the record is decrypted, the MAC is verified and the data is decompressed. The data is proceeded to the higher layer protocol.

2. The encrypted message of the SSL record can be either a message of the SSL Handshake Protocol, or a message of the SSL Change Cipher Spec Protocol, or a message of the SSL Alert Protocol, or a message with application data. This depends on the protocol which is above the SSL record protocol in the processing stack. The type of message is indicated in the Content Type field of the SSL record header.

Solution for Assignment 14:

In the SSL Handshake Protocol the cryptographic parameters (algorithms, keys) used to secure a connection between a client and a server are negotiated. The protocol can be viewed as having four phases.

In the **first phase security capabilities are established**. The phase is initiated by the client and is used to initiate a logical connection by exchanging a `Hello` messages between the client and the server. When a client first connects to a server it is required to send the `Client Hello` as its first message. The client can also send a `Client Hello` in response to a `Hello Request` message that comes from the server or on its own initiative in order to renegotiate the security parameters in an existing connection. The server processes the `Client Hello` message and responds with either a `Handshake Failure` (fatal error will occur and the connection will fail) or `Server Hello` message. With the `Client Hello` and `Server Hello` messages the security enhancement capabilities (protocol version, session ID, cipher suite, and compression method) between client and server are established.

The **second phase** of the handshake protocol is the **Authentication and Key Exchange**. If the server is to be authenticated (which is generally the case), it sends its certificate in `Certificate` message immediately following the Server Hello message. If the server has no certificate, or has certificate only used for signing, or Fortezza/DMS key exchange is used, the server sends the `Server Key Exchange` message. With this message, the server informs the client about its public key parameters (for RSA, for Diffie-Hellman or for the Fortezza key exchange algorithm).
When the server is authenticated, it may request a certificate from the client if that is appropriate to the cipher suite selected, and send the `Certificate Request` message. The final message in phase 2, and the one that is always required, is the `Server Hello Done` message sent by the server. This indicates that the `Server Hello` and associated messages phase of the handshake is complete. The server will then wait for a client response.

The **third phase** of the Handshake is the **client authentication and key exchange**. Upon receipt of `Server Hello Done` message, the client should verify that the server provided a valid certificate if required and check that the `server hello` parameters are acceptable. If all is satisfactory, the client sends one or more messages back to the server. If the server has sent a `Certificare Request` message, the first message the client sends is its `Certificate` message (or a `No Certificate` alert, if no suitable certificate is available). Next is the `Client Key Exchange` message, and the content of this message depends on the type of key exchange selected before. Then the client may send a `Certificate Verify` message to provide explicit verification of the client certificate (if the client has sent a certificate).

The **fourth phase** completes the setting up of a secure connection. The client sends a `Change Cipher Spec` message and copies the pending `Cipher Spec` into the current `Cipher Spec`. The client then immediately sends the `Finished` message to verify that the key exchange and authentication process were successful. This message includes hash values of all handshake messages starting at `Client Hello` up to the `Finished` message and is the first message protected with the negotiated algorithms and keys. In response of these two messages, the server sends its own `Change Cipher Spec` message, transfers the pending to the current `Cipher Spec`, and then sends its own `Finished` message under the new `Cipher Spec`. Recipients of `Finished` message must verify that the contents are correct. No acknowledgement of the Finished message is required.

At this point, the handshake is complete and the client and server may begin to exchange application layer data, which are carried by the SSL record layer.

Solution for Assignment 15:

Examples for sites where the SSL protocol is used to protect the data:

- www.amazon.de
- www.sparkasse-hagen.de

- www.quelle.de

Examples for Sites without SSL protection:

- www.lob.de
- www.versicherungsvergleich.de

Solution for Assignment 16:

Contents and locations of the `.htaccess` files:

1. Case 1
   - Location: `assignment`, content:
     ```
     AuthName "assignment"
     AuthUserFile /etc/apache/passwd
     AuthGroupFile /etc/apache/htgroup
     AuthType Basic
     <LIMIT GET PUT>
     require user cryptoad
     </LIMIT>
     ```
   - Location: `course`, content:
     ```
     AuthName "course"
     AuthUserFile /etc/apache/passwd
     AuthGroupFile /etc/apache/htgroup
     AuthType Basic
     <LIMIT GET PUT>
     require group all
     </LIMIT>
     ```
   - Location: `cu1`, content:
     ```
     AuthName "cu1"
     AuthUserFile /etc/apache/passwd
     AuthGroupFile /etc/apache/htgroup
     AuthType Basic
     <LIMIT GET PUT>
     require user cryptoad charly
     </LIMIT>
     ```
   - Location: `cu2`, content:
     ```
     AuthName "cu2"
     AuthUserFile /etc/apache/passwd
     AuthGroupFile /etc/apache/htgroup
     AuthType Basic
     <LIMIT GET PUT>
     require group students2
     require user cryptoad
     </LIMIT>
     ```
2. Case 2
   - Location: `assignment`, content:

```
AuthName "assignment"
AuthUserFile /etc/apache/passwd
AuthGroupFile /etc/apache/htgroup
AuthType Basic
<LIMIT GET PUT>
require group all
</LIMIT>
```

- Location: `ea1`, content:

```
AuthName "ea1"
AuthUserFile /etc/apache/passwd
AuthGroupFile /etc/apache/htgroup
AuthType Basic
<LIMIT GET PUT>
require user cryptoad
</LIMIT>
```

- Location: `ea3`, content:

```
AuthName "ea3"
AuthUserFile /etc/apache/passwd
AuthGroupFile /etc/apache/htgroup
AuthType Basic
<LIMIT GET PUT>
require user cryptoad
</LIMIT>
```

Solution for Assignment 17:

The data moves in the following manner:

- If you type the URL of a web site into your browser, your browser sends a request to the web site for the page. For example, if you type the URL http://www.amazon.com into your browser, your browser will contact Amazon's server and request its home page.
- When the browser does this, it will look on your machine for a cookie file that Amazon has set. If it finds an Amazon cookie file, your browser will send all of the name-value pairs in the file to Amazon's server along with the URL. If it finds no cookie file, it will send no cookie data.
- Amazon's web server receives the cookie data and the request for a page. If name-value pairs are received, Amazon can use them.
- If no name-value pairs are received, Amazon knows that you have not visited their site before. The server creates a new ID for you in Amazon's database and then sends name-value pairs to your machine in the header for the web page it sends. Your machine stores the name- value pairs on your hard disk.
- The web server can change name-value pairs or add new pairs whenever you visit the site and request a page.

There are other pieces of information that the server can send with the name-value pair. One of these is an expiration date. Another is a path (so that the site can associate different cookie values with different parts of the site).

Solution for Assignment 18:

Web sites use cookies in many different ways. Here are some of the most common examples:

- Sites can accurately determine how many readers actually visit the site. It turns out that because of proxy servers, caching, concentrators and so on, the only way for a site to accurately count visitors is to set a cookie with a unique ID for each visitor. Using cookies, a site can:
  - Determine how many visitors arrive
  - Determine how many are new vs. repeat visitors
  - Determine how often a visitor has visited.

  The way the site does this is by using a database. The first time a visitor arrives, the site creates a new ID in the database and sends the ID as a cookie. The next time the user comes back, the site can increment a counter associated with that ID in the database and know how many times that visitor returns.
- Sites can store user preferences so that the site can look different for each visitor (often referred to as customisation). For example, if you visit msn.com, it offers you the ability to change content/layout/color. It also allows you to enter your zip code and get customized weather information. Most sites seem to store preferences like this in the site's database and store nothing but an ID as a cookie, but storing the actual values in name-value pairs is another way to do it (we'll discuss why this approach has lost flavour below).
- Ecommerce Sites can implement things like shopping carts and "quick check-out" options. The cookie contains an ID and lets the site keep track of you as you add different things to your cart. Each item you add to your shopping cart is stored in the site's database along with your ID value. When you check out, the site knows what is in your cart by retrieving all of your selections from the database. It would be impossible to implement a convenient shopping mechanism without cookies or something like it.

Solution for Assignment 19:

a) At least one of the two mixes M_1, M_2 must be trustworthy.

b) Without consideration of the addresses's encryption, the incoming message to the mix M_1 is

$$\begin{aligned} c_1 = E_{k_{e,M_1}}(E_{k_{e,M_2}}(E_{k_{e,B}}(m))) &= E_{k_{e,M_1}}(E_{k_{e,M_2}}(m^{e_B} \mod n_B) \\ &= E_{k_{e,M_1}}(E_{k_{e,M_2}}(124^5 \mod 28829) \\ &= E_{k_{e,M_1}}(E_{k_{e,M_2}}(11695)) \\ &= E_{k_{e,M_1}}(11695^{3674911} \mod 6012707) \\ &= E_{k_{e,M_1}}(625690) \\ &= 625690^5 \mod 55465219 \\ &= 7226241. \end{aligned}$$

The outcoming message from M_1 is

$$\begin{aligned} D_{k_{d,M_1}}(c_1) &= D_{k_{d,M_1}}(7226241) \\ &= 7226241^{d_{M_1}} \mod n_{M_1} \\ &= 7226241^{44360237} \mod 55465219 \\ &= 625690. \end{aligned}$$

Solution for Assignment 20:

If more than one message is sent over the DC network, then this causes a collosion of messages. That is, the messages would not be correctely sent. A common key establishment is not practicable in a DC network (in its basic version) since a large portion of keys, having the same length as the messages, must be permanently exchanged. In this case, the time delay of the network's traffic will be increased as well as the required bandwith.

Solution for Assignment 21:

1. Stateful packet filtering maintains state information about passed IP packets. Stateful inspection peeks into the payload date where transport and application layer data usually appears, to identify connections or virtual circuits.

2. The following problems exist with stateful packet filtering:

   a. In stateful packet filtering the problem remains, that there is no association between a data stream and a preciously authenticated and authorized user.

   b. The packet filter has to keep track of the state. This increases the load on the packet filter and opens it to denial of servive attacks.

   c. Packets may be denied when they should have been accepted.

3. There exist some security related advantages in the use of NAT:

   a. NAT helps to enforce the packet filter and firewall control over outbound connections.

b. NAT can help to restrict incoming traffic, if dynamic allocation schemes are used.

c. NAT helps to conceal the internal networks' configuration.

Solution for Assignment 22:

Recall Example 6.1-2. Suppose our intranet has the subnet IP number `172.16.1.0` with subnet mask `255.255.255.0`. The users in the local subnet are not allowed to access any services on the Internet. The users on the Internet are allowed to access the local WWW server on the machine with IP number `172.16.1.10`.

The commands to build a packet filter with `iptables` on a router or dual-homed host between the Internet and the local subnet are as follows:

At first the kernel module that provides support for `netfilter` has to be loaded into the kernel:

```
$ modprobe iptables
```

The commands to build the chains for the `filter` table, that fulfill the above policy are:

```
$ iptables -F OUTPUT
$ iptables -P OUTPUT DROP
$ iptables -F INPUT
$ iptables -P INPUT DROP
$ iptables -F FORWARD
$ iptables -P FORWRAD DROP
$ iptables -A FORWARD -m tcp -p tcp -s 172.16.1.10/32
    --sport 80 -d 0/0 --syn -j DROP
$ iptables -A FORWARD -m tcp -p tcp -s 0/0
    -d 172.16.1.10/0 --dport 80 -j ACCEPT
$ iptables -A FORWARD -m tcp -p tcp -s 172.16.1.10/32
    --sport 80 -d 0/0 -j ACCEPT
```

Solution for Assignment 23:

There is no general recommendation which layer of the Internet model is best suited to provide the security for Internet services and applications. The best layer (to be chosen for providing security of an application) depends on the security services required and the application environment in which the services must be deployed. Note that a higher level of security of some services can be reached by the use of multiple layers rather than only one. The transport layer is more appropriate to provide security for applications of the electronic commerce field (using TLS or SSL). Email service which provides a point-to-point communication is richly secured above the application layer (with S/MIME specifications or PGP programs). Bulk data cryptographic transformations, such as message authentication, integrity checking, and data encryption are well performed at the network layer.

Solution for Assignment 24:

1. A distributed file system with NFS works following the client-server model. A client's user can access the server's file system in a transparent way on the basis of the Remote Procedure Calls (RPCs). RPC guarantees platform independence, i.e. the client and the server do not need to exist on the same physical system. In turn, RPC uses data types defined by the eXternal Data Representation (XDR) to hide all details of architecture-dependent data representation.

2. The NFS-request from a client to a server as well as the datacontent will be transmitted using the TCP or UDP protocol, i.e. the confidentiality is not guaranteed. The requirement that only authorized users are able to read or manipulate files, is not adequately met by NFS. AUTH UNIX and AUTH DES are two of many mechanisms for authenticating clients and servers that apply to NFS. The AUTH UNIX follows the UNIX style for authentication. Thus, it provides no real authentication. The AUTH DES authentication which uses the Deffie-Hellman algorithm to derive a DES key for encrypting the communication between client and server, has some security weaknesses. On one hand, the DES is already decrypted. On the other hand, the choice of small modulus in the Deffie-Hellman algorithm leads to the decryption the NFS packets.

3. The use of AFS (Andrew File System) and CFS (Cryptographic File System) enables a secured distributed file system. AFS uses Kerberos for the mutual authentication of client and server. To provide authorization service, AFS uses special Access Control Lists (ACL).

   CFS pushes encryption services into the UNIX file system such as NFS. CFS enables to encrypt files when

   - they are stored on a local harddisk,
   - they are sent to a remote file server,
   - they are stored on a remote file server, or
   - backups of them are generated.

Solution for Assignment 25:

1. S/MIME provides the following cryptographic security services for electronic messaging applications: authentication, message integrity and non-repudiation of origin (using digital signatures) and privacy and data security (using encryption).

2. A MIME entity must be prepared for signing or enveloping, i.e. it must be canonicalized and encoded for transfer.

3. Two formats for signed messages are defined by S/MIME:

`application/pkcs7-mime` with `signedData` as parameter, and `multipart/signed`.

By the `multipart/signed` format, a recipient without any S/MIME or CMS processing facilities is able to read the message. A recipient of a message signed using `signedData` format, cannot read the message if he possess no S/MIME facilities.

Solution for Assignment 26:

1. Examples of executable content which can affect the network security are binary mail attachments, helper applications and plug-ins, scripting languages, Java applets, and ActiveX.

2. A denial of service attack can be mounted against a host, when a JavaScript code is written in a malicious manner. For example, the JavaScript code can require too much of the CPU or memory resources, or it can try to open a large number of windows on the victim's screen.

3. Java uses some techniques to limit the risk related to a downloading of Java applets. These techniques are the Java Sandbox, the SecurityManager class, the Bytecode Verifier, and the Java Class Loader.

   - Java Sandbox: enables Java programs to run only on a JVM inside a restricted runtime environment.

   - SecurityManager class: will be called before any unsafe operation is executed.

   - Java Class Loader: examines classes to ascertain that they do not violate the runtime system.

   - Bytecode Verifier: assures whether a downloaded bytecode stems from a valid Java program.

Solution for Assignment 27:

The following three security Services are provided by GSM:

- **Authenticity** is realized by a permanent secret key K_i for each subscriber.

- **Confidentiality of user-related data** is realized by enciphering with the A5 algorithm.

- **Confidentiality of the user identity** is realized by the use of *Temporary Mobile Subscriber Identities (TMSI)*.

Solution for Assignment 28:

In Section 8.2.4 the basic procedure for GSM authentication is presented. For this solution, especially for the second answer, we consider a procedure in some more detaile.

1. Entities involved in GSM authentication are:

   SIM:*Subscriber Identity Module*

   BTS:*Base Transceiver Station*

   VLR / MSC:*Visitor Location Register / Mobile Switching Centre (for circuit services)*

   HLR / AC:*Home Location Register / Authentication Centre*

2. The method for authentication employed between HLR/AC and SIM is a challenge-response mechanism using cryptographic secure 128-bit random numbers (`RAND`) and a 128-bit authentication key (K_i), which is kept both in the SIM card and the *Authentication Centre*.

   1. The *Mobile Station* signs into the network.
   2. The *Mobile Switching Centre* requests five triplets from the *Home Location Register*.
   3. The *Home Location Register* creates the five requested triplets. Each of these triplets consists of the following three values:
      - A non-predictable 128-bit random number (`RAND`).
      - A 32-bit Singed Response (`SRES`), calculated form `RAND` and K_i by utilizing the authentication algorithm `A3`.
      - A 64-bit Session Key (K_c), calculated form `RAND` and K_i by utilizing the key generation algorithm `A8`. After a successful authentication the Session Key is used to encrypt the subsequent communication with the `A5` algorithm.
   4. The five triplets are sent from the *Home Location Register* to the *Mobile Switching Centre*.
   5. The *Mobile Switching Centre* relays the random number (`RAND`) from the first triplet to the *Base Transceiver Station*.
   6. The *Base Transceiver Station* relays this random number to the *Mobile Station*.
   7. The *Mobile Station* utilizes the `A3` algorithm to calculate the Signed Response (`SRES'`) from the received random number `RAND` and the stored Authentication Key K_i.
   8. The *Mobile Station* sends the Signed Response (`SRES'`) to the *Base Transceiver Station*.
   9. The *Base Transceiver Station* relays the Signed Response (`SRES'`) to the *Mobile Switching Centre*.

10. The *Mobile Switching Centre* compares `SRES` from the first triplet with `SRES′`.

11. Only if `SRES` is equal to `SRES′` access to the network is granted to the *Mobile Station*.

3. The network authenticates the SIM. Therefore the SIM has to demonstrate its knowledge of K_i. The authentication of the network by the SIM is not provided by GSM.

Solution for Assignment 29:

Examples for GSM attacks are:

SMS spoofing attack: SMS spoofing is a very simple attack. With an apropriate application the originating address of an SMS can be freely set to any desired set of alphanumeric characters.

This attack compromises the security goals of authenticity and nonrepudiation.

False base station attack: Base stations in GSM do not authenticate themselves to mobile stations. Therefore an attacker can set up a false base station and then force a mobile station to turn off encryption in order to eavesdrop the connection. See [Mitchell01].

The false base station attack compromises the security goal of confidentiality.

IBM side channel partitioning attacks: These attacks developed by IBM [Rao02] do not attack the cryptographic algorithms directly, but exploit the large quantities of sensitive information emanating from the side-channels such as power consumption and electromagnetic radiations. Therefore it is possible to find out the subscriber authentication key K_i and clone the SIM card in a relatively short period of time.

The IBM side channel partitioning attacks compromise the security goals of authenticity and authorisation.

Solution for Assignment 30:

- Authentication of the *Home Location Register* (**HLR**) to the *User Services Identity Module* (**USIM**).
- Integrity on the air interface both for data and for signalling information.
- Key freshness.
- New algorithms.
- Key length of 128 Bit.
- Core network signalling security.
- Encryption ends in the *Radio Network Controllers* (**RNC**).
- Anonymity

Solution for Assignment 31:

a) Man-in-the-middle attacks (e.g. false base station attacks as described in the solutions for exercises in course unit 9) are prevented by the mutual authentication provided in the UMTS standard.

b) In order to facilitate the evolution from GSM to UMTS, mobile devices allow a UMTS subscriber to roam between GSM and UMTS networks. This means that a UMTS subscriber can connect to a serving network with UMTS backbone components via a GSM base station subsystem. In this case an attacker can perform a man-in-the-middle attack since GSM base stations do not support integrity protection, which is necessary for the network authentication defined in the UMTS standard (see [Meyer04]).

Solution for Assignment 32:

Examples for WAP attacks are:

Truncation attack: Some of the alert messages used in the WAP protocol are sent in cleartext and are not properly authenticated. It is possible for an attacker to replace an encrypted datagram with a plaintext alert message with the same sequence number. Thus arbitrary packets can be truncated from the data stream.

This attack compromises the security goals of authenticity and integrity.

XOR MAC and stream ciphers: The WTLS protocol supports a 40-bit XOR MAC to provide message integrity. But in combination with stream ciphers as used in WAP an XOR MAC does not provide any message integrity because if a bit at position n in the ciphertext is invertet the MAC can be made to match easily by inverting the bit ($n \bmod 40$) in the MAC.

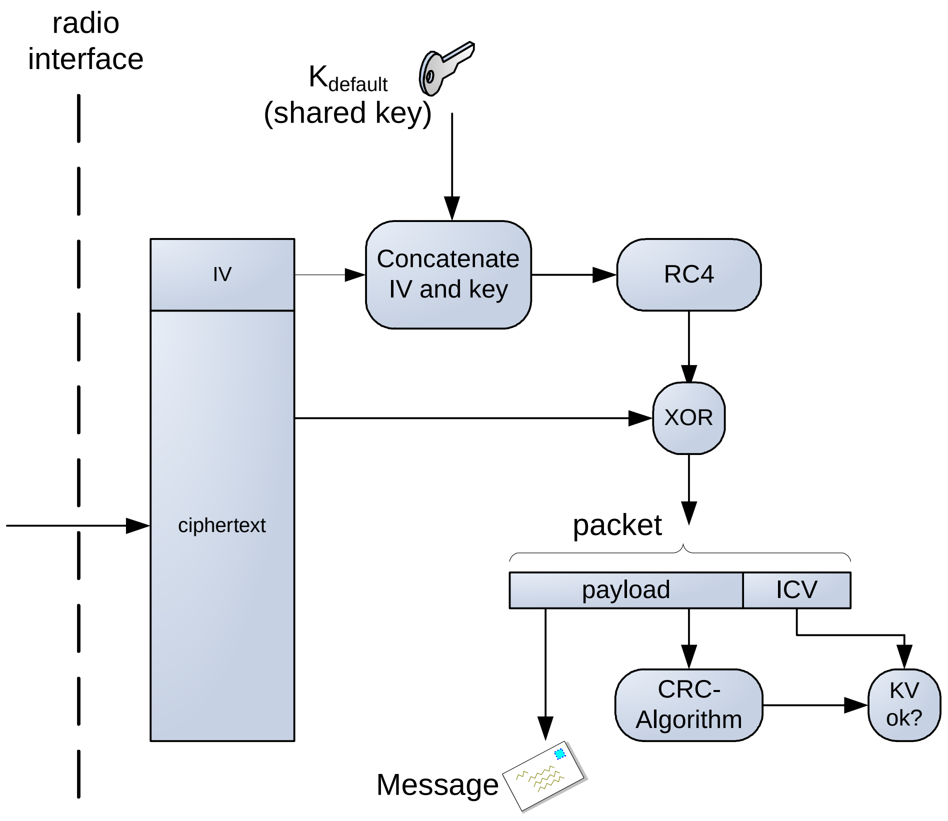
This attack compromises the security goal of integrity.

Solution for Assignment 33:

In a plaintext attack the attacker knows the plaintext and the corresponding ciphertext. In the challenge-response phase of the *Shared Key Authentication*, the challenge (plaintext) and the response (ciphertext) are both transmitted with no encryption.

Solution for Assignment 34:

The decryption engine looks similar to the encryption part:



Solution for Assignment 35:

Examples for WAP attacks are:

Truncation attack: Some of the alert messages used in the WAP protocol are sent in cleartext and are not properly authenticated. It is possible for an attacker to replace an encrypted datagram with a plaintext alert message with the same sequence number. Thus arbitrary packets can be truncated from the data stream.

This attack compromises the security goals of authenticity and integrity.

XOR MAC and stream ciphers: The WTLS protocol supports a 40-bit XOR MAC to provide message integrity. But in combination with stream ciphers as used in WAP an XOR MAC does not provide any message integrity because if a bit at position n in the ciphertext is invertet the MAC can be made to match easily by inverting the bit (n mod 40) in the MAC.

This attack compromises the security goal of integrity.

Solution for Assignment 36:

An electronic payment system is a system providing services for paying goods and information electronically over insecure open networks. Usually, diverse cryptographic technologies and network security tools are used to ensure that network users can make payments fast and safely.

The main categories of electronic payment systems are the following:

- Credit Card-based payment systems

- Electronic check systems
- Electronic cash systems
- Micropayment systems

Credit Card-based systems have to transmit customers credit card numbers over insecure networks. That is the reason, why Credit Card-based systems need much stronger security than other electronic payment systems.

Solution for Assignment 37:

1. *OIsalt* is a nonce which is used when creating a hash of the order description to prevent dictionary attacks. This nonce can prevent an attack from guessing the hash value, *H(OIData)*, by trying all possible combinations of dictionary words in the order description.

2. The main feature of electronic check payment systems is the immediate transfer from the payers account to the payee´s account at the time of purchase. The NetBill server is the bottleneck of the system because the NetBill server has to be involved into all transactions.

3. The main parties involved in an electronic cash system are:

   - Consumer
   - Merchant
   - Issuing bank
   - The trusted third party (optional)

   The following protocols are the main protocols contained in an electronic cash system:

   - Registration protocol (set up account with the bank)
   - Withdrawal protocol
   - Payment protocol
   - Deposit protocol
   - Tracing protocol

   Payer_code is generated secretly by the user. A hash of it, *H(payer_code)*, is included in the payment information which enables the user to prove the bank later, after the coin has been deposited, that he made the payment.

4. Electronic micropayment systems are systems that can provide the transfer of very small amounts of money in a single transaction while minimizing the communication traffic. The main point of an electronic micropayment systems is that, when the cost of commiting a fraud exceeds the value of the purchase,

then the system is thought to be secure. The computation consuming cryptography primitives will not used in electronic micropayment systems.

ID# is a unique identifier of the scrip in the Millicent system as a serial number. It is used to prevent double spending of the scrip. Furthermore, some parts of it are used to select the secret which is called master_scrip_secret. The master_scrip_secrets are known to the vendor only.

Solution for Assignment 38:

It is a program that can autonomously migrate to visited hosts in the heterogeneous network for performing tasks on behalf of its owner.

Solution for Assignment 39:

A malicious host can attack a mobile agent on data and service. In case of data, it can modify data by deletion, alteration, and modification. However, in case of service, it performs denial of service (i.e. does not let the mobile agent migrate to other visited hosts, does not provide its resources, repeatedly executes mobile agent).

Solution for Assignment 40:

There are four main researches for the mobile agent paradigm namely application, infrastructure, security, and standard.

Solution for Assignment 41:

There are two possibilities namely: dialogue and message passing. The proposed paradigm should be equipped with a mobile agent locating function, a standard message format and a reliable message delivery protocol. In order to develop a mobile agent locating function, the proposed solution must equip each mobile agent a unique name and location update scheme. About a reliable message delivery protocol, the proposed paradigm should be able to guarantee correct message delivery to the target agent. For a standard message format, it ensures the format of an exchanging message and how a message should be interpreted.

Solution for Assignment 42:

a) 1. The result of the scan is shown below:

```
Starting Nmap 3.9999 ( http://www.insecure.org/nmap ) at 2006-05-16 15:51
Westeuropäische Normalzeit

Interesting ports on Bitburger.fernuni-hagen.de (132.176.12.10):
(The 1664 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 3.9p1 (protocol 2.0)
25/tcp   open  smtp        Postfix smtpd
80/tcp   open  http        Apache httpd 1.3.33 ((Unix) PHP/4.3.9)
111/tcp  open  rpcbind      2 (rpc #100000)
139/tcp  open  netbios-ssn Samba smbd (workgroup: LGKS)
443/tcp  open  ssl         OpenSSL
```

```
3306/tcp open  mysql       MySQL (unauthorized)
5432/tcp open  postgresql  PostgreSQL DB

MAC Address: 00:02:B3:95:4C:FB (Intel)
Device type: general purpose
Running: Linux 2.4.X
OS details: Linux 2.4.19 - 2.4.20
Uptime 27.059 days (since Wed Apr 19 14:27:12 2006)
Service Info: Host:  bitburger.fernuni-hagen.de
Nmap finished: 1 IP address (1 host up) scanned in 14.109 seconds
```

2. As can be seen from the result the following ports are open:
   - 22 (SSH)
   - 25 (SMTP)
   - 80 (HTTP)
   - 111 (RPCbind)
   - 139 (NetBios)
   - 443 (SSL)
   - 3306 (Mysql database server)
   - 5432 (postgresSql database server)
     None of them is useless.
3. The bitbuger server runs a Linux 2.4 version.

b) 1. The following protocols can be identified from the trace
   - Ethernet II
   - ARP (Address Resolution Protocol)
   - IPX (Internet Packet eXchange)
   - PIM (Protocol Independent Multicast)
   - IP (Internet Protocol)
   - TCP (Transport Control Protocol)
   - UDP (User Datagram Protocol)
   - HTTP (HyperText Transport Protocol)
   - NTP (Network Time Protocol).
2. The hexadecimal values of the source address is 00 07 ec 13 f7 fc and the hexadecimal value of the destination address is 00 e0 18 e9 fd 93 for packet 30.
3. The fields of the IP and TCP header can be determined from the capture and are:
   1. **IP header**
      IP Version = 4

IP header length = 20

Differentiated Services Code Point-DSCP = 0

ECN (Explicit Congestion Notification) = 0

Total length = 796 bytes

Identification = 22121

Flags

- reserved bit: not set
- don´t fragment (DF): set, since the DF-bit is set the packet can not be fragmented
- more fragment (MF): not set

Fragment offset = 0

TTL = 63

Protocol = tcp

Header checksum = 59dc (the packet has been transmitted without error)

Source IP address = 132.176.114.181

Destination IP address =132.176.12.129

2. **TCP header**

Source port 80 (standard HTTP port)

Destination port (1529)

Sequence number = 10950 (relative sequence number)

Acknowledment number = 722 (relative acknowlegment number)

Flags: PSH and ACK are set, the rest is not set

Window size = 49640

Checksum = 5467 (hexadeximal)

4. The packet consists of 756 bytes of payload + 20 bytes (TCP header) + 20 bytes (IP header) + 14 Bytes (Ethernet Header) = 810 Bytes.

5. The method used to request the information is `GET`. As can be seen from the URI a Graphic Interchange Format (GIF) image has been requested from the client with the IP number 132.176.12.129 (packet 93). The response came from the server with the IP address 132.176.114.181 (packet *95*). The URI is `/info/images/pfeil_rot.gif`.

6. An Apache 1.3.33, PHP 5.0.5 server is used and it runs HTTP/1.1, status code: 200, which means the information requested has been retrieved without problems.

7. Average packet size = 662 bytes, average byte/s = 9813,676 bytes/s, total amount of TCP packets = 86 and UDP packets = 10.

Solution for Assignment 43:

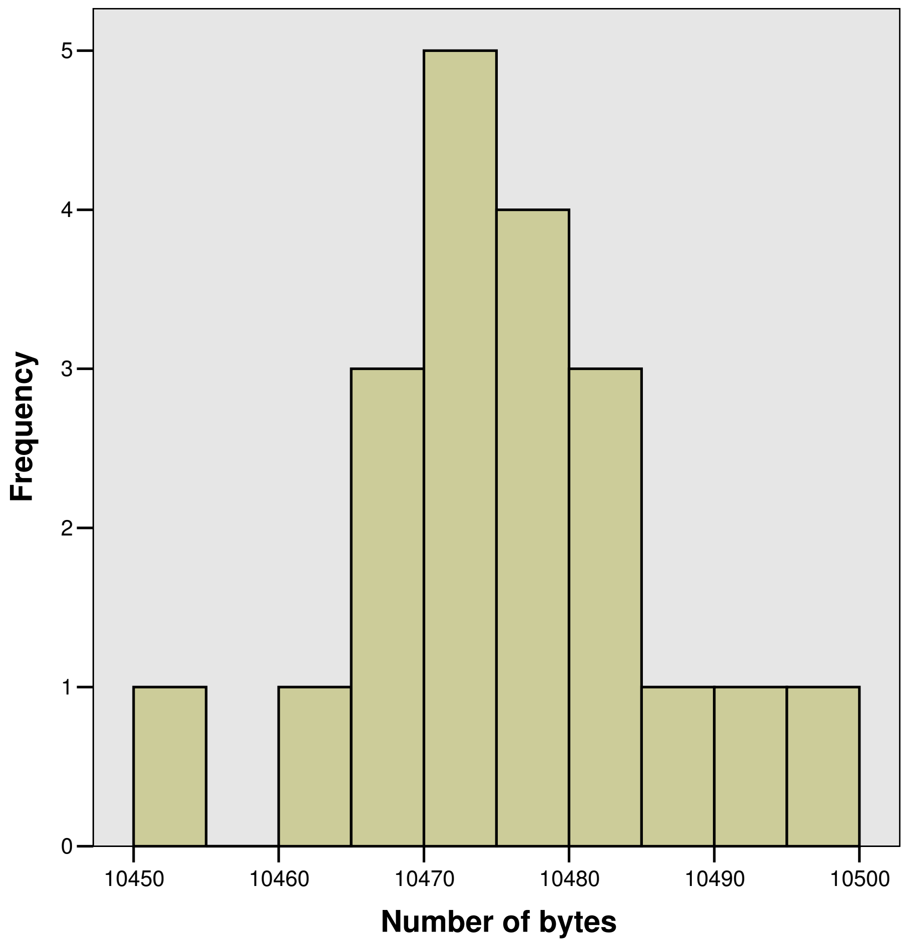
a) The frequency table for the number of bytes received is illustrated below:

	Frequency	Percent	Cumulative Percent
10454	1	5	5
10460	1	5	10
10465	1	5	15
10467	1	5	20
10469	1	5	25
10471	1	5	30
10472	3	15	45
10473	1	5	50
10475	1	5	55
10476	2	10	65
10478	1	5	70
10480	1	5	75
10483	1	5	80
10484	1	5	85
10486	1	5	90
10492	1	5	95
10498	1	5	100
Total	20	100	

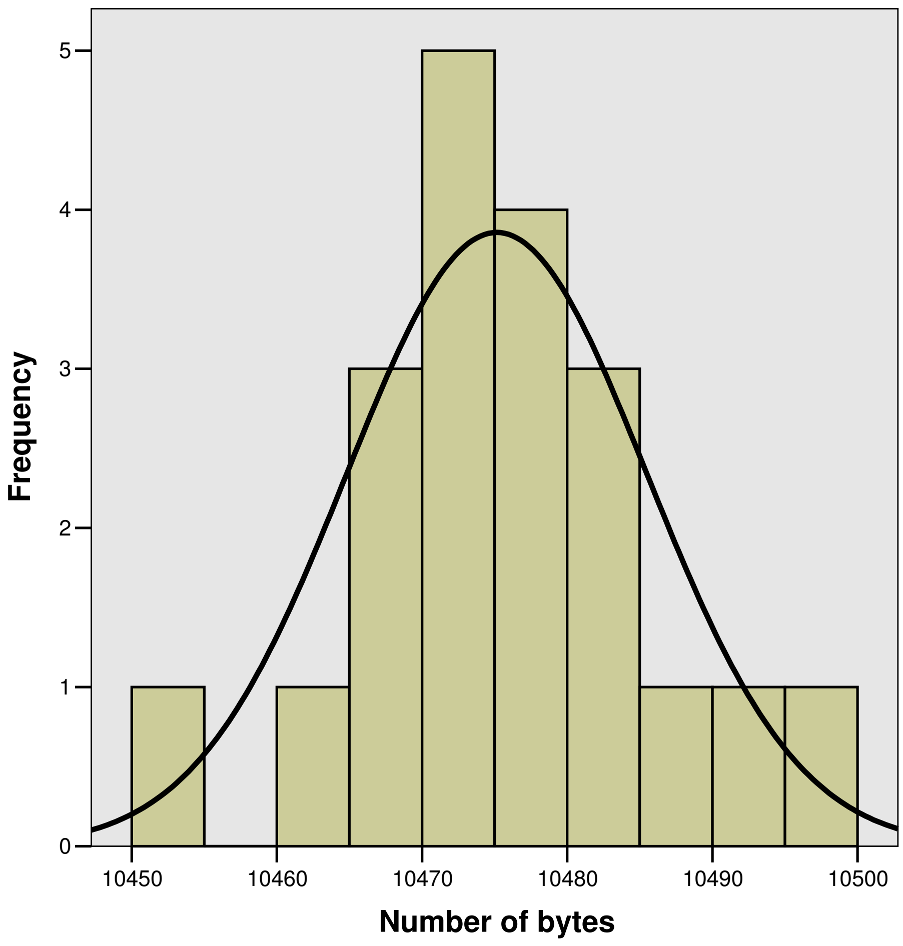
The frequency table for the number of packets received, is shown below:

	Frequency	Percent	Cumulative Percent
10	1	5	5
11	4	20	25
12	7	35	60
13	8	40	100
Total	20	100	

b) The histogram of the number of bytes received is shown below:



c) The parameters of the normal distribution are $\mu = 10475,15$ and $\sigma = 10,343$. The approximation is shown below. As can be seen from the picture, this normal distribution does not approximate the histogram well and therefore can not be used to detect anomalies. A transformation function has to be applied to the data set in Table 1 to generate a data set which will better fit a normal distribution. The confidence interval for a level of significance $\alpha = 0.05$ is $\$\mu \pm 2\sigma = [10454,46 \quad 10495,836]$.


Frequency
0
1
2
3
4
5
10450
10460
10470
10480
10490
10500
Number of bytes

d) The Matrix of observations M is :

$$M = \begin{bmatrix} 13 & 10498 \\ 12 & 10478 \\ 11 & 10480 \\ 13 & 10476 \\ 12 & 10454 \\ 12 & 10472 \\ 12 & 10486 \\ 13 & 10483 \\ 11 & 10492 \\ 12 & 10467 \\ 13 & 10484 \\ 13 & 10475 \\ 13 & 10476 \\ 13 & 10465 \\ 12 & 10460 \\ 11 & 10469 \\ 12 & 10473 \\ 11 & 10472 \\ 10 & 10472 \\ 13 & 10471 \end{bmatrix}$$

e) The mean vector $\bar{X}$ equals

$$\bar{X} = \begin{pmatrix} 12,1 \\ 10475,15 \end{pmatrix}$$

$$X_1 = \begin{pmatrix} 13 \\ 10498 \end{pmatrix}$$

$$(X_1 - \bar{X})(X_1 - \bar{X})^t = \begin{bmatrix} 0,81 & 20,56 \\ 20,56 & 522,12 \end{bmatrix}$$

f) The Hotelling's test can be calculated for both events (observations).

The Hotelling´s test for x_{20}

$$T^2_{x_{20}} = (0,9 \quad 22,85) \begin{bmatrix} 0,83 & 1,09 \\ 1,09 & 106,98 \end{bmatrix}^{-1} \cdot \begin{pmatrix} 0,9 \\ 22,85 \end{pmatrix} = 5,42$$

This value is then multiplied with $\frac{n \cdot (n-p)}{p \cdot (n+1) \cdot (n-1)}$ hereby $n = 20$ and $p = 2$. The result is $2,445$. The tabulated value for $p = 2$ and $n - p = 18$ for a level of significance $\alpha = 0.05$ is $3,55$. Since the calculated value is smaller than the tabulated value, x_{20} is not an anomaly.

The same is done for x_{21}

$$T^2_{x_{21}} = (-0,1 \quad 2,85\ ) \begin{bmatrix} 0,83 & 1,09 \\ 1,09 & 106,98 \end{bmatrix}^{-1} \cdot \begin{pmatrix} -0,1 \\ 2,85 \end{pmatrix} = 0,096$$

The value is then multiplied by 0.451. The result is 0.043 smaller than the tabulated value. Therefore x_{21} is not an anomaly.

Solution for Assignment 44:

a) 1.
```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-ATTACKS /usr/bin/id command attempt";
flow:to_server,established;
content:"/usr/bin/id"; nocase;
classtype:web-application-attack; sid:1332; rev:5;)
```

This rule tells Snort to generate an alert message whenever an attacker tries to execute the Unix command `"id"` to gather information about users or group on the host. Snort generates an alert whenever a TCP packet carrying web traffic is observed coming from `$EXTERNAL_NET` headed to `$HTTP_SERVER` containing the command string `"/usr/bin/id"`. Hereby the content is not case sensitive. The variables `$EXTERNAL_NET` and `$HTTP_SERVER` are to be defined in the configuration file `snort.conf`. Snort attacks are categorized along classes and are assigned priorities. This attack belongs to the web application attacks class and the priority is high. It is therefore an attack which should be taken very seriously. Further information related to the attack can be found under `reference` and with the `sid`, each Snort rule can be uniquely identified. The `rev` is used for rules update.

2.
```
alert tcp $EXTERNAL_NET any -> $HOME_NET 22
(msg:"EXPLOIT ssh CRC32 overflow NOOP";
flow:to_server,established;
content:"|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|";
reference:bugtraq,2347; reference:cve,2001-0144;
reference:cve,2001-0572; classtype:shellcode-detect;
sid:1326; rev:6;)
```

In this rule Snort generates an alert message whenever an attacker tries to perform a buffer overflow on a SSH server and execute arbitrary code with the privilege of the user running the daemon (root). Snort generates an alert whenever a TCP packet is observed coming from `$EXTERNAL_NET` headed to `$HOME_NET` containing the binary string `"|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|"`. The pipe indicates that this string is a binary one. This attack belongs to the shell-code detect class and the priority is high. `flow`: indicates that rules are only applied to clients or servers, in this case on established server connections.

3.
```
alert tcp $SQL_SERVERS 3306 -> $EXTERNAL_NET any
(msg:"MYSQL server greeting"; flow:from_server,established;
content:"|00|"; depth:1; offset:3;
flowbits:set,mysql.server_greeting; flowbits:noalert;
reference:bugtraq,10655;
reference:url,www.nextgenss.com/advisories/mysql-authbypass.txt;
classtype:attempted-user; sid:3665; rev:1;)
```

   This rule generates an alert message when an attacker tries to bypass password authentication in MySQL 4.1. The offset specifies the number of bytes to be skipped and the depth tells Snort how far it should look in the packet. In this rule the first 3 bytes are skipped and the the binary content `"|00|"` is searched for in the first byte of the payload.

4.
```
alert ip $EXTERNAL_NET any -> $HOME_NET any
(msg:"BAD-TRAFFIC Unassigned/Reserved IP protocol"; ip_proto:>134;
reference:url,www.iana.org/assignments/protocol-numbers;
classtype:non-standard-protocol; sid:1627; rev:3;)
```

   This rule generates an alert message when an attacker tries to use unassigned or reserved fields in the IP packet header to send crafted packets. In IPv4 the field protocol is 8 bit in length, that means that 256 values can be used. The values of the field are specified in RFC 1700. In this rule, Snort tests all values which are greater than 134. The attack belongs to the class non-standard-protocol and the priority is medium.

5.
```
alert tcp $EXTERNAL_NET any -> $HOME_NET 79
(msg:"BACKDOOR CDK"; flow:to_server,established;
content:"ypi0ca"; depth:15; nocase; reference:arachnids,263;
classtype:misc-activity; sid:185; rev:5;)
```

   This rule generates an alert message when an attacker tries to connect to a victim server, where a backdoor has been installed by issuing the password `"ypi0ca"`. The attack can be detected by searching for the string `"ypi0ca"` in the first 15 bytes of the packet. The attack belongs to the class misc-activity and priority is low.

Solution for Assignment 45:

a) 1. A rule to identify this attack is given below:

```
alert tcp $EXTERNAL_NET any → $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-IIS cmd32.exe access"; content:"cmd32.exe"; nocase;)
```

2.
```
alert tcp any any → any 80
(msg "Code Red Worm", content: "default.ida?"; nocase;)
```

3.
```
alert tcp $EXTERNAL_NET any -> $HOME_NET any
(flags: SF;msg:"SYN-FIN Scan";)
```

4.
```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-ATTACKS /etc/shadow access";
flow:to_server,established; content:"/etc/shadow"; nocase;)
```

References

References for Chapter "Introduction"

[Buc99] Johannes Buchmann. *Einführung in die Kryptographie.* Springer Verlag, 1999.

[CB94] William R. Cheswick and Steven M. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker.* Addison-Wesley, 1994.

[Com00] Douglas E. Comer. *The Internet Book.* Prentice Hall International, 2000.

[Den87] Dorothy E. Denning. IEEE Transactions on software engineering, Vol. Se-13, No. 2, FEBRUARY 1987, 222-232.

[DH76] W. Diffie and Martin E. Hellman. *New directions in cryptography.* IEEE Transactions on Information Theory, 22(6):644– 654, November 1976.

[DH99] Naganand Doraswamy and Dan Harkins. *IPSec.* Prentice Hall International, 1999.

[Eck06] Claudia Eckert. *IT-Sicherheit: Konzepte, Verfahren, Protokolle 4. überarbeitete und erweiterte Auflage.* R. Oldenbourg Verlag, 2006

[Fei99] Sidnie Feit. *TCP/IP - Architecture, Protocols and Implementation with IPv6 and IP security.* Mc Graw Hill, 1999.

[FRU00] Stephan Fischer, Christoph Rensing, and Rödig Utz. *Open Internet Security.* Springer Verlag, 2000.

[Fuh98] Kai Fuhrberg. *Internet-Sicherheit: Browser, Firewalls und Verschlsselung.* Hanser Verlag, 1998.

[Kad91] Firoz Kaderali. *Digitale Kommunikationstechnik (Band 1).* Vieweg Verlag, 1991.

[Kad95] Firoz Kaderali. *Digitale Kommunikationstechnik (Band 2).* Vieweg Verlag, 1995.

[Kad00] Firoz Kaderali. *Anonymität im Internet* Berichte aus der Kommunikationstechnik Band5, Shaker Verlag, Aachen, 2000.

[Kah67] David Kahn. *The Codeberakers.* Macmillan Publishing Company, New York, 1967.

[KCL+00] Firoz Kaderali, Biljana Cubaleska, Bernhard Löhlein, Sonja Schaup, and Oliver Stutzke. *Course - Foundations of Cryptology.* Department of Communication Systems, University of Hagen, 2000.

[KDS+00] Firoz Kaderali, Thomas Demuth, Dagmar Sommer, Gerd Steinkamp, and Michael Stepping. *Course 20018 - Internet Techniques*. Department of Communication Systems, University of Hagen, 2000.

[Los99] Pete Loshin. *TCP/IP clearly explained*. Morgan Kaufmann, 1999. 3rd Edition.

[MOV96] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[Opp00] Ralf Oppliger. *Security Technologies for the Internet*. Artech House Publishers, 2000.

[Sch96] Bruce Schneier. *Applied Cryptography: protocols, algorithms, and source code in C*. John Wiley and Sons, 1996.

[Sch97] C. L. Schuba. *On the Modeling, Design, and Implementation of Firewall Technology*. PhD thesis, Pudue University, December 1997.

[Sha49] Claude Shannon. *Communication theory of secrecy systems*. Bell System Technical Journal, 28:656–715, 1949.

[Smi97] Richard E. Smith. *Internet cryptography*. Addison Wesley Longman Inc., 1997.

[Spu00] Charles E. Spurgeon. *Ethernet: The Definitive Guide*. O'Reilly and Associates, 2000.

[Ste98] Lincoln D. Stein. *Web Security: A Step-by-Step Reference Guide*. Addison Wesley, 1998.

[Sti95] Douglas R. Stinson. *Cryptography: Theory and Practice*. CRC Press, 1995.

[Tan00] Andrew S. Tanenbaum. *Computernetzwerke*. Pearson Studium, 2000.

References for Chapter "Basics"

[Buc99] Johannes Buchmann. *Einführung in die Kryptographie*. Springer Verlag, 1999.

[Cha92] Lyman Chapin. *The internet standards process*. Technical report, RFC (Request for Comments) 1310, Status: Informational, IETF (Internet Engineering Task Force), 1992. http://www.ietf.org/rfc/rfc1310.txt.

[DH98] S. Deering, and R. Hinden.*Internet protocol, version 6 (ipv6) specification*. Technical report, RFC (Request for Comments) 2460, Status: Draft Standard, IETF (Internet Engineering Task Force), 1998. http://www.ietf.org/rfc/rfc2460.txt.

[ISO84] ISO 7498-1. *Information processing systems – open systems interconnection – basic reference model - part 1:* The basic model. Technical report, International Organization for Standardization, Geneva, Switzerland, 1984. ISO 7498-1, first edition 1984, second edition 1994.

[ISO89] ISO 7498-2. *Information processing systems – open systems interconnection – basic reference model - part 2:* Security architecture. Technical report, International Organization for Standardization, Geneva, Switzerland, 1989. ISO 7498-2, first edition 1989.

[KDS+00] Firoz Kaderali, Thomas Demuth, Dagmar Sommer, Gerd Steinkamp, and Michael Stepping. *Course 20018 - Internet Techniques.* Department of Communication Systems, University of Hagen, 2000.

[MF00] David A. McGrew and Scott R. Fluhrer. *Attacks on additive encyrption of redundant plaintext and implications on internet security.* In *Seventh Workshop on Selected Areas in Cryptography (SAC'00), Lecture Notes in Computer Science.* Springer-Verlag, 2000.

[Pos80] J. Postel. *User datagram protocol.* Technical report, RFC (Request for Comments) 768, Status: Standard, IETF (Internet Engineering Task Force), 1980. http://www.ietf.org/rfc/rfc0768.txt.

[Pos81a] J. Postel. *Internet control message protocol.* Technical report, RFC (Request for Comments) 792, Status: Standard, IETF (Internet Engineering Task Force), 1981. http://www.ietf.org/rfc/rfc0792.txt.

[Pos81b] J. Postel. *Internet protocol.*Technical report, RFC (Request for Comments) 791, Status: Standard, IETF (Internet Engineering Task Force), 1981. http://www.ietf.org/rfc/rfc0791.txt.

[Pos81c] J. Postel. *Transmission control protocol.* Technical report, RFC (Request for Comments) 793, Status: Standard, IETF (Internet Engineering Task Force), 1981. http://www.ietf.org/rfc/rfc0793.txt.

[RP94] J. Reynolds and J. Postel. *Assigned numbers.* Technical report, RFC (Request for Comments) 1700, Status: Standard, IETF (Internet Engineering Task Force), 1994. http://www.ietf.org/rfc/rfc1700.txt.

[Sch96] Bruce Schneier. *Applied Cryptography: protocols, algorithms, and source code in C.* John Wiley and Sons, 1996.

[ZCC00] Elisabeth D. Zwicky, Simon Cooper, and D. Brent Chapman. *Building Internet Firewalls.* O'Reilly and Associates, 2000. Second Edition.

References for Chapter "Internet Security Protocols"

[DA99] T. Dierks and C. Allen. The tls protocol, version 1.0. Technical report, RFC (Request for Comments) 2246, Status: Informational, IETF (Internet Engineering Task Force), 1999. http://www.ietf.org/rfc/rfc2246.txt.

[DH99] Naganand Doraswamy and Dan Harkins. *IPSec*. Prentice Hall International, 1999.

[Eck04] Claudia Eckert. *IT-Sicherheit*.R. Oldenbourg Verlag, 2004

[Fei99] Sidnie Feit. *TCP/IP - Architecture, Protocols and Implementation with IPv6 and IP security.* Mc Graw Hill, 1999.

[FRU00] Stephan Fischer, Christoph Rensing, and Rödig Utz. *Open Internet Security.* Springer Verlag, 2000.

[KCL+00] Firoz Kaderali, Biljana Cubaleska, Bernhard Löhlein, Sonja Schaup, and Oliver Stutzke. *Course - Foundations of Cryptology.* 2000. Department of Communication Systems, University of Hagen, http://etonline.fernuni-hagen.de/lehre/rub/rub1.

[KDS+00] Firoz Kaderali, Thomas Demuth, Dagmar Sommer, Gerd Steinkamp, and Michael Stepping. *Course 20018 - Internet Techniques. 2000.* Department of Communication Systems, University of Hagen, http://www.ice-bachelor.fernuni-hagen.de.

[KPS95] Charlie Kaufman, Radia Perlman, and Mike Speciner. *Network Security: Private Communication in a Public World.* Prentice Hall, 1995.

[Opp00] Ralf Oppliger. *Security Technologies for the Internet.* Artech House Publishers, 2000.

[RUB97] Aviel Rubin. *Web Security Sourcebook.* John Wiley and Sons, 1997.

[Smi97] Richard E. Smith. *Internet cryptography.* AddisonWesley Longman Inc., 1997.

[Sta96] William Stallings. *Practical Cryptography for Data Internetworks.* IEEE Computer Society Press, 1996.

[Sta99] William Stallings. *Cryptography and Network Security.* Prentice Hall, 1999. Second Edition.

[Ste98] Lincoln D. Stein. *Web Security: A Step-by-Step Reference Guide.* Addison Wesley, 1998.

References for Chapter "World Wide Web Security"

[ABH97] Derek Atkins, Paul Buis, and Chris Hare. *Internet Security, Professional Reference Guide, 2nd Edition.* New Riders Publishing, 1997.

[Fla96] David Flanagan. Javascript: *The Definitive Guide.* O'Reilly & Associates, 1996.

[Fla99] David Flanagan. *JAVA in a Nutshell, 3rd Edition.* O'Reilly & Associates, 1999.

[Opp00] Ralf Oppliger. *Security Technologies for the Internet.* Artech House Publishers, 2000.

[Rub97] Aviel Rubin. *Web Security Sourcebook.* John Wiley and Sons, 1997.

[Ste98] Lincoln D. Stein. *Web Security: A Step-by-Step Reference Guide.* Addison Wesley, 1998.

References for Chapter "Anonymity Techniques"

[Chaum81] David L. Chaum. Untraceable electronic mail, return adress, and digital pseudonyms. *Communication of the ACM*, 24(2):84–88, 1981.

[Chaum88] David Chaum. The dining cryptographers problem: Unconditionel sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[Demuth98] Thomas Demuth and Andreas Rieke. Anonym im World Wide Web? JANUS-Schutz von Inhalteanbietern im WWW. *DuD Datenschutz und Datensicherheit*, 22(11):623–627, 1998.

[Demuth01] Thomas Demuth and Andreas Rieke. Janus: Server anonymity in the world wide web. *10th Annual EICAR Conference, 2nd European Anti-Malware Conference*, 2001.

[Franz97] Elke Franz, Anja Jerichow, and Andreas Pfitzmann. Systemisierung und Modellierung von Mixen. *Verlliche ITSysteme, GI-Fachtagung VIS '97, DuD Fachbeiträge*, pages 172–190, 1997.

[Franz98] Elke Franz, A. Graubner, A. Jerichow, and A. Pfitzmann. Modelling mix-mediated anonymous communication and preventing pool-mode attacks. *Global IT Security, Proc. of IFIP/SEC'98, 14th Intern. Information Security Conference, Schriftenreihe der Österreichischen Computer Gesellschaft, Band 116*, pages 554–560, 1998.

[Pfitzmann87] Andreas Pfitzmann and Michael Waidner. Networks without user observability. *Computer and Security*, 6(2):158–66, 1987.

References for Chapter "Packet filters and firewall systems"

[Bel94] Steven M. Bellovin. Firewall-friendly FTP. Technical report, RFC (Request for Comments) 1579, Status: Informational, IETF (Internet Engineering Task Force), February 1994. http://www.ietf.org/rfc/rfc1579.txt.

[CB94] William R. Cheswick and Steven M. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker.* Addison- Wesley, 1994.

[CB00] William R. Cheswick and Steven M. Bellovin. *Internet security: Firewalls and gateways.* Addison-Wesley, 2000.

[KD00] Olaf Kirch and Terry Dawson. *Linux Network Administrator's Guide.* O'Reilly and Associates, 2000. Second Edition.

[KK92] D. Koblas and M.R. Koblas. Socks. In N. [N.92], pages 77–82.

[Lee96] M. Leech. Username/Password Authentication for SOCKS V5. Technical report, RFC (Request for Comments) 1929, Status: Proposed Standard, IETF (Internet Engineering Task Force), March 1996. http://www.ietf.org/rfc/rfc1929.txt.

[LGL+96] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. SOCKS Protocol Version 5. Technical report, RFC (Request for Comments) 1928, Status: Proposed Standard, IETF (Internet Engineering Task Force), March 1996. http://www.ietf.org/rfc/rfc1928.txt.

[McM96] P. McMahon. GSS-API Authentication Method for SOCKS Version 5. Technical report, RFC (Request for Comments) 1961, Status: Proposed Standard, IETF (Internet Engineering Task Force), 1996. http://www.ietf.org/rfc/rfc1961.txt.

[N.92] N. N., editor. *Proceedings of the Third Annual USENIX Security Symposium.* IEEE Press, September 1992.

[Opp00] Ralf Oppliger. *Security Technologies for the Internet.* Artech House Publishers, 2000.

[RM+96] Y. Rekhter, , B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address Allocation for Private Internets. Technical report, RFC (Request for Comments) 1918, Status: Best Current Praxis, IETF (Internet Engineering Task Force), February 1996. http://www.ietf.org/rfc/rfc1918.txt.

[Sch97] C. L. Schuba. *On the Modeling, Design, and Implementation of Firewall Technology.* PhD thesis, Pudue University, December 1997.

[SE01] P. Srisuresh and K. Egevang. Traditional IP Network Address Translator (Traditional NAT). Technical report, RFC (Request for Comments) 3022, Status: Informational, IETF (Internet Engineering Task Force), January 2001. http://www.ietf.org/rfc/rfc3022.txt.

[SH99] P. Srisuresh and M. Holdrege. IP Network Address Translator (NAT) Terminology and Considerations. Technical report, RFC (Request for Comments) 2663, Status: Informational, IETF (Internet Engineering Task Force), August 1999. http://www.ietf.org/rfc/rfc2663.txt.

[TS00] G. Tsirtsis and P. Srisuresh. Network Address Translation - Protocol Translation (NAT-PT). Technical report, RFC (Request for Comments) 2766, Status: Proposed Standard, IETF (Internet Engineering Task Force), February 2000. http://www.ietf.org/rfc/rfc2766.txt.

[ZCC00] Elisabeth D. Zwicky, Simon Cooper, and D. Brent Chapman. *Building Internet Firewalls.* O'Reilly and Associates, 2000. Second Edition.

References for Chapter "Application Layer Security (Chapter 7.1 - 7.6)"

[Cro82] David H. Crocker. STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES. RFC (Request for Comments) 822, Status: Proposed Standard, IETF (Internet Engineering Task Force), 1982. http://www.ietf.org/rfc/rfc822.txt.

[ea95] J. Galvin et al. Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted. RFC (Request for Comments) 1847, Status: Proposed Standard, IETF (Internet Engineering Task Force), 1995. http://www.ietf.org/rfc/rfc1847.txt.

[ea96a] N. Freed et al. Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples . RFC (Request for Comments) 2049, Status: Proposed Standard, IETF (Internet Engineering Task Force), 1996. http://www.ietf.org/rfc/rfc2049.txt.

[ea96b] N. Freed et al. Multipurpose Internet Mail Exten- sions (MIME) Part Four: Registration Procedures. RFC (Request for Comments) 2048, Status: Proposed Standard, IETF (Internet Engineering Task Force), 1996. http://www.ietf.org/rfc/rfc2048.txt.

[ea96c] N. Freed et al. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. RFC (Request for Comments) 2045, Status: Proposed Standard, IETF (Internet Engineering Task Force), 1996. http://www.ietf.org/rfc/rfc2045.txt.

[ea96d] N. Freed et al. Multipurpose Internet Mail Exten- sions (MIME) Part Two: Media Types. RFC (Request for Comments) 2046, Status: Proposed Standard, IETF (Internet Engineering Task Force), 1996. http://www.ietf.org/rfc/rfc2046.txt.

[ea98] J. Callas et al. OpenPGP Message Format. RFC (Request for Comments) 2440, Status: Proposed Standard, IETF (Internet Engineering Task Force), November 1998. http://www.ietf.org/rfc/rfc2440.txt.

[FRU00] Stephan Fischer, Christoph Rensing, and Rödig Utz. *Open Internet Security*. Springer Verlag, 2000.

[GS95] Simson Garfinkel and Gene Spafford. *Web security & Commerce*. O'Reilly & Associates, Inc., 1995. First Edition.

[Hug95] Larry J. Hughes and Jr. *Actually Useful Internet Security Techniques*. New Riders Publishing, 1995.

[Hou99] B. Kaliski. PKCS #7: Cryptographic Message Syntax Version 1.5. RFC (Request for Comments) 2315, Status: Proposed Standard, IETF (Internet Engineering Task Force), 1998. http://www.ietf.org/rfc/rfc2315.txt.

[KCL+00] Firoz Kaderali, Biljana Cubaleska, Bernhard LÄohlein, Sonja Schaup, and Oliver Stutzke. *Course - Foundations of Cryptology*. 2000. Department of Communication Systems, University of Hagen, http://etonline.fernuni-hagen.de/lehre/rub/rub1.

[KDS+00] Firoz Kaderali, Thomas Demuth, Dagmar Sommer, Gerd Steinkamp, and Michael Stepping. *Course 20018 - Internet Techniques*. 2000. Department of Communication Systems, University of Hagen, http://www.ice-bachelor.fernuni-hagen.de.

[Moo96] K. Moore. MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text. RFC (Request for Comments) 2047, Status: Proposed Standard, IETF (Internet Engineering Task Force), 1996. http://www.ietf.org/rfc/rfc2047.txt.

[N98] N. N. System Management Guide: Communications and Networks. 1998. http://www.rs6000.ibm.com/doc_link/en_US/a_doc_lib/aixbman/commadmn/toc.htm.

[N01] N. N. eMail Wiretapping Exploit. 2001. http://home.de.netscape.com/de/info/security-doc.html.

[Opp97] Ralf Oppliger. *Internet and Intranet Security*. Artech House Publishers, 1997.

[Opp00] Ralf Oppliger. *Security Technologies for the Internet*. Artech House Publishers, 2000.

[PR85] J. Postel and J. Reynolds. FILE TRANSFER PROTOCOL (FTP). RFC (Request for Comments) 595, Status: Proposed Standard, IETF (Internet Engineering Task Force), 1985. http://www.ietf.org/rfc/rfc595.txt.

[Ram99a] B. Ramsdell. S/MIME Version 3 Certificate Handling. RFC (Request for Comments) 2632, Status: Proposed Standard, IETF (Internet Engineering Task Force), 1999. http://www.ietf.org/rfc/rfc2632.txt.

[Ram99b] B. Ramsdell. S/MIME Version 3 Message Speci¯catim. RFC (Request for Comments) 2633, Status: Proposed Standard, IETF (Internet Engineering Task Force), 1999. http://www.ietf.org/rfc/rfc2633.txt.

[SM87] Inc. Sun Microsystems. XDR: External Data Representation Standard. RFC (Request for Comments) 1014, Status: Proposed Standard, IETF (Internet Engineering Task Force), 1987. http://www.ietf.org/rfc/rfc1014.txt.

[SM88] Inc. Sun Microsystems. RPC: Remote Procedure Call Protocol Specification Version 2. RFC (Request for Comments) 1057, Status: Proposed Standard, IETF (Internet Engineering Task Force), 1988. http://www.ietf.org/rfc/rfc1057.txt.

[SM89] Inc. Sun Microsystems. NFS: Network File System Protocol Specification. RFC (Request for Comments) 1094, Status: Proposed Standard, IETF (Internet Engineering Task Force), 1989. http://www.ietf.org/rfc/rfc1094.txt.

[ZCC00] Elisabeth D. Zwicky, Simon Cooper, and D. Brent Chapman. *Building Internet Firewalls.* O'Reilly and Associates, 2000. Second Edition

References for Chapters "Application Layer Security and Security in Wireless and Mobile Networks (Chapter 7.7 - 8.2)"

[Borcherding] Malte Borcherding. Mobile security - an overview of GSM, SAT and WAP. pages 133–141.

[GSM0320] GSM 03.20 (ETS 300 534). *Digital cellular telecommunications system (phase 2); Security related functions.* Technical report.

[GSM1203] GSM 12.03 (ETS 300 614). *Digital cellular telecommunications system (phase 2); Security management.* Technical report.

[Golic97] Jovan Dj. Golic. *Cryptanalysis of alleged A5 stream cipher.* pages 239–255.

[Kaderali00b] Firoz Kaderali, Biljana Cubaleska, Bernhard Löhlein, Sonja Schaup, and Oliver Stutzke. *Course - Foundations of Cryptology.* 2000. Department of Communication Systems, University of Hagen

[Schneier96] Bruce Schneier. *Applied Cryptography: protocols, algorithms, and source code in C.* John Wiley and Sons, 1996.

[Shamir99] Adi Shamir. *Real time cryptanalysis of A5/1 on a PC.* Technical report, Weizmann Institute of Technology, Israel, 1999.

[Mitchell01] C. Mitchell. *The security of the GSM air interface protocol.* Technical Report RHUL-MA-2001-3, Royal Holloway, University of London, 2001.

[Rao02] J. Rao, P. Rohatgi, H. Scherzer and S. Tinguely. *Partitioning Attacks: Or How to Rapidly Clone Some GSM Cards.* Proceedings of the IEEE Symposium on Security and Privacy, Oakland, 2002

References for Chapter "Security in Wireless and Mobile Networks (Chapter 8.3 - 8.4)"

[3GP02a] 3GPP. 3g security; network domain security; ip network layer security. Technical report, 3rd Generation Partnership Project, June 2002. 3G TS 33210 V5.1.0.

[3GP02b] 3GPP. 3g security; network domain security; map application layer security. Technical report, 3rd Generation Partnership Project, 3 2002. 3G TS 33200 V5.0.0.

[3GP02c] 3GPP. Specification of the 3gpp confidentiality and integrity algorithms; document 1: f8 and f9 specification. Technical report, 3rd Generation Partnership Project, June 2002. 3GPP TS 35201 V5.0.0.

[3GP02d] 3GPP. Specification of the 3gpp confidentiality and integrity algorithms; document 2: Kasumi specification. Technical report, 3rd Generation Partnership Project, June 2002. 3GPP TS 35202 V5.0.0.

[DU99] DTS/TSGS-033120U. *3g security; security principles and objectives.* Technical report, 3rd Generation Partnership Project, 1999. 3G TS 33120 V3.0.0.

[Fox02a] Dirk Fox. *Bluetooth security.* DuD, Datenschutz und Datensicherheit. October 2002.

[Heijden00] Marcel Van der Heijden and Marcus Taylor. *Understanding WAP Wirless; Applications, Devices, and Services.* Artech House, 2000.

[Kaderali00b] Firoz Kaderali, Biljana Cubaleska, Bernhard Löhlein, Sonja Schaup, and Oliver Stutzke. *Course - Foundations of Cryptology.* 2000. Department of Communication Systems, University of Hagen

[Loehlein02] Bernhard Loehlein. *Bluetooth - technology, security and weaknesses.* Proceedings of SCI 2002.

[Meyer04] Ulrike Meyer, Susanne Wetzel. *A Man-in-the-Middle Attack on UMTS.* Proceedings of WiSe 2004

[Puetz01a] Stephan Pütz, Roland Schmitz, and Tobias Martin. Security mechanisms in umts. *Datenschutz und Datensicherheit (DuD)*, pages 323–332, June 2001.

[Stepping02] Christoph Stepping. *Integration of security aspects in umts and bran*. Institute for data communication systems, university of siegen. 2002.

[Tzvetkov] Vesselin Tzvetkov and Biljana Cubaleska. Wap protocol security solutions for mobile commerce. Proceedings of Sci2002, Volume IV.

[Wap01a] *Wap architecture*. July 2001. WAP-210-WAPArch-20010712. http://www.wapforum.org/.

[Wap01b] *Wireless identity module*. July 2001. Version 12-July-2001. http://www.wapforum.org/.

[Wap01c] *Wireless session protocol specification*. July 2001. Approved Version 5-July-2001. http://www.wapforum.org/.

[Wap01d] *Wireless transaction protocol*. July 2001. Version 10-Jul-2001. http://www.wapforum.org/.

[Wap01e] *Wmlscript crypto library*. June 2001. Version 20-Jun-2001. http://www.wapforum.org/.

References for Chapter "Security in Wireless and Mobile Networks (Chapter 8.5 - 8.6)"

[Dent] Donald J. Dent. The bluetooth wireless revolution. Proceedings of Sci2002, Volume IV.

[EA05] Jon Edney and William A. Arbaugh: *Real 802.11 Security: Wi-Fi Protected Access and 802.11i*, Addison Wesley, 2005

[FMS01] Scott Fluhrer, Itsik Martin and Adi Shamir: *Weaknesses in the Key Scheduling Algorithm of RC4*, August 2001

[Fox02a] Dirk Fox. *Bluetooth security*. DuD, Datenschutz und Datensicherheit. October 2002.

[Http] http://www.bluetooth.com.

[Loehlein02] Bernhard Loehlein. *Bluetooth - technology, security and weaknesses*. Proceedings of SCI 2002.

[Re04] Jörg Rech: *Wireless LANs*, Heise Zeitschriften Verlag, 2004

[RFC2898] *PKCS #5: Password Based Cryptography*, Version 2, http://www.ietf.org/rfc/rfc2898.txt

References for Chapter "Electronic Payment Systems"

[Adachi03] N.Adachi, S.Aoki, Y.Komano, and K.Ohta. The security problems of rivest and shamir’s payword scheme. In *CEC’2003*, pages 20–24. IEEE Computer society Press, 2003.

[Asokan97] N. Asokan, Phillipe A. Janson, Michael Steiner, and Michael Waidner. *The state of the art in electronic payment systems*. IEEE Computer, 30(9):28–35, 1997.

[Bellare95] Mihir Bellare, Juan Garay, Ralf Hauser, Amir Herzberg, Hugo Krawczyk, Michael Steiner, Gene Tsudik, and Michael Waidner. iKP – A family of secure electronic payment protocols. In *Proceedings of the 1st USENIX Workshop on Electronic Commerce*, pages 89–106, 1995.

[Boly94] J-p. Boly and et al. The esprit project cafe. In *ESORICS’94*, volume 875 of *Lecture Notes in Computer Science*, pages 217– 230. Springer Verlag, 1994.

[Brands93] Stefan Brands. *An efficient off-line electronic cash system based on the representation problem*. Technical Report CSR9323, Centrum voor Wiskunde en Informatica, Computer Science Department of Algorithmics and Architecture, 1993.

[Brands94] Stefan Brands. Untraceable off-line cash in wallets with observers. In *CRYPTO’93*, volume 773 of *Lecture Notes in Computer Science*, pages 302–318, Berlin, 1994. Springer Verlag.

[Chaum83] David Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: CRYPTO ’82*, pages 199–203, 1983.

[Chaum85] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030, 1985.

[Chaum90] D. Chaum and M. Naor A. Fiat. Untraceable electronic cash. In *CRYPTO ’88*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327. Springer Verlag, 1990.

[Chaum91] D. Chaum and E. van Heyst. *Group signatures*. volume 547, pages 257–265, 1991.

[Chaum92] David Chaum. *Achieving electronic privacy*. Scientific American, 267(2):96, 1992.

[Chaum93] David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In *CRYPTO ’92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer Verlag, 1993.

[Cox95] Tygar D. Cox, B. and M. Sirbu. *Netbill security and transaction protocol.* Technical report, Carnegie Mellon University, 1995.

[Eastlake96] D. Eastlake 3rd, B. Boesch, S. Crocker, and M. Yesil. Cybercash credit card protocol version 0.8. In *Internet Network Working Group, Request for Comments: RFC 1898*, 1996.

[Ferguson94] Neils Ferguson. Extensions of single-term coins. In *CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 292–301, Berlin, 1994. Springer Verlag.

[Franklin93] M.Franklin and M.Yung. Secure and efficient off-line electronic digital money. In *20th Int. Colloquium on Automata, Languages and programming (ICALP), volume 700 of Lecture Notes in Computer Science*, pages 265–276. Springer Verlag, 1993.

[FST] http://echeck.commerce.net/overview/consortium/fstc.html.

[Glassman95] S. Glassman, M. S. Manasse, M. Abadi, P. Gauthier, and P. Sobalvarro. The millicent protocol for inexpensive electronic commerce. In *Proceedings of Fourth International World Wide Web Conference*, Dec 1995.

[ITU97] ITU-T X.509 V3 (formerly CCITT X.509). *The Directory: Authentication Framework.* June 1997.

[Kocker97] P. Kocker, A. Freier, and P. Karlton. *The SSL Protocol Version 3.0*. Netscape Communications Corp., March 1997.

[Kügler01] D.Kügler and H.Vogt. Fair tracing without trustees. In *Financial Cryptography 2001*, volume 2339 of *Lecture Notes in Computer Science*, pages 136–148. Springer Verlag, 2001.

[MC96a] MasterCard and VISA Corporations. *Secure Electronic Transaction(SET) Specification - Book 1: Business Description*. June 1996.

[MC96b] MasterCard and VISA Corporations. *Secure Electronic Transaction(SET) Specification - Book 2: Programmer's Guide*. June 1996.

[MC96c] MasterCard and VISA Corporations. *Secure Electronic Transaction(SET) Specification - Book 3: Formal Protocol Definition*. June 1996.

[Medvinsky93] Gennady Medvinsky and B. Clifford Neuman. Netcash: A design for practical electronic currency on the internet. In Proceedings of the First ACM Conference on Computer and Communications Security, pages 102–106, 1993.

[Mondex] http://www.mondex.com

[Neuman95] B. Clifford Neuman and Gennady Medvinsky. Requirements for network payment: The netcheque perspective. *In Proceedings of IEEE COMPCON'95*, pages 32–36, 1995.

[Pfitzmann00] B. Pfitzmann and A. Sadeghi. *Self-escrowed cash against user blackmailing*. volume 1962, pages 42–45, 2000.

[Qiu02] Weidong Qiu, Kefei Chen, and Dawu Gu. A new off-line privacy protecting e-cash system with revokable anonymity. In *International Security Conference, ISC'98, Sao Paulo, Brazil, volume 2433 of Lecture Notes in Computer Science*, pages 177–190. Springer Verlag, 2002.

[Rivest96] Ronald L. Rivest and Adi Shamir. Payword and micromint: Two simple micropayment schemes. In *Security Protocols Workshop*, pages 69–87, 1996.

[Sander99] T. Sander and A. Ta-Shma. *Auditable, anonymous electronic cash*. volume 1666, pages 555–572, 1999.

[Semper] http://www.semper.org.

[Sirbu95] M. Sirbu and J.D. Tygar. Netbill: An electronic commerce system optimized for network delivered services. In *Proceedings of IEEE COMPCON'95*, pages 20–25, 1995.

[Sirbu96] M. Sirbu and J. Chuang Public-key based ticket granting service in kerberos8. In *Internet Network Working Group, Request for Comments: RFC 1510*, May 1996.

[Solms92] S.von Solms and D.Naccache. On blind signatures and perfect crime. Computer and Security, 11(6):581, 1992.

[Steiner88] J. G. Steiner, B. C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. In *Proceedings of USENIX* winter, pages 191–202, 1988.

[Traoré99] J. Traoré. Group signature and their relevance to privacyprotecting off-line electronic cash systems. In *ACISP99*, volume 1587, pages 228–243, 1999.

[Wayner97] Peter Wayner. *Digital cash: Commerce on the net*. Morgan Kaufmann; 2nd edition, London, 1997.

[Yang03] C-N.Yang and H.-T.Teng. An efficient method for finding minimum hash chain of multi-payword chains in micropayment. In *CEC'2003*, pages 45–49. IEEE Computer society Press, 2003.

References for Chapters "Security Aspects in Mobile Agents Systems and Copyright Protection"

[Alfalayleh04] M. Alfalay, L. Brankovic. *An Overview of Security Issues and Technique in Mobile Agents.* CMS Conference on Communications and Multimedia Security 2004.

[Beirman02] E. Beirman, E. Cloete. *Classification of Malicious Host Threats in Mobile Agent Computing.* SAICSIT 2002.

[Breugst98] D. Milojicic, M. Breugst, I. Busse, J. Campbell, S. Covaci, B. Friedman, K. Kosaka, D. Lange, K. Ono, M. Oshima, C. Tham, S. Virdhagriswaran and J. White. *MASIF The OMG Mobile Agent System Interoperability Facility.* Proceedings of the International Workshop on Mobile Agents (MA ’98), Stuttgart, September 1998.

[Carzaniga97] A. Carzaniga, G. P. Picco, G. Vigna. *Designing Distributed Applications with Mobile code paradigm.* Proceedings of the 19th International Conference on Software Engineering,1997.

[Chess95] D. Chess, B.Grosof, C. Harrison, D. Levine, C. Parris, G. Tsudik. *Itinerant Agents for Mobile Computing.* IEEE Personal Communications. 1995.

[Collberg02] C. Collberg, C. Thomborson. *Watermarking, Tamper-Proofing, and Obfusication - Tools for Software Protection.* IEEE Transactions on Software Engineering,Volume 28, Issue 8, Aug. 2002 Page(s):735-746.

[Das03] S. Das, K. Shuster, C. Wu.*Dynamic Information Retrieval Optimization Using Mobile Agents.* AAMAS 2003.

[Derbas04] G. Derbas, A. Kayssi, H. Artail. A. Chehab. *TRUMMAR - A Trust Model for Mobile Agent Systems Based on Reputation.* IEEE/ACS international conference on pervasive services, 2004.

[Esparza03] O. Esparza, M. Fernandez, M. Soriano. J. L. Munoz, and J. Forne. *Mobile Agent Watermarking and Fingerprinting: Tracing Malicious host.* DXEA 2003, LNCS 2736.

[Farmer96] W. M. Farmer, J. D. Guttman, V. Swarup. *Security for Mobile Agents: Authentication and State Apprisal.* EROSIC 1996.

[Ferrer01] J. D. Ferrer. *Mobile Agent Route Protection through Hash-Based Mechanism.* INDOCRYPT 2001, LNCS 2247.

[FIPA02] *http://www.fipa.org.*

[FM96] J.S. Fritzinger and M. Mueller. *Java Security.* Sun Microsystems, Inc., 1996.

[Fuggetta98] A. Fuggetta, G. P. Picco, G. Vigna. *Understanding Code Mobility.*IEEE Transactions on Software Engineering, 1998.

[Gelbart04] O. Gelbart. *Mobile Agent Security: Protecting The Host Platform form Malicious Mobile Agents.* 2004.

[Guttman98] R. H. Guttman, A.G. Moukas, P. Maes. *Agent-mediated Electronic Commerce: A Survey.* 1998.

[Hohl98] F. Hohl. *Time Limited BlackBox Security: Protecting Mobile Agents From Malicious Hosts.* LNCS 1419, 1998.

[Janzen00] W. Jansen, T. Karygiannis. *NIST Special Publication 800-19 Mobile Agent Security.* National Institute of Standard and Technology, 2000.

[Janzen00b] W. Jansen. *Countermeasure for mobile agent security.*National Institute of Standard and Technology, 2000.

[Jennings98] . R. Jennings, M. J. Wooldridge. *Agent Technology.*

[Karnik98] N. M. Karnik, A. R. Tripathi. *Design Issues in Mobile Agent Programming System.* Department of Computer Science, Univ. of Minnesota, 1998.

[Karjoth98] G. Karjoth, N. Asokan, C. Gülcü. *Protecting the Computation Results of Free-Roaming Agents.* Proceedings of the 2nd International Workshop on Mobile Agents. LNCS, Vol. 1477. Springer-Verlag, Berlin Heidelberg New York (1998), 195–207.

[Kruegel01] C. Kruegel, T. Toth. *Applying Mobile agent Technology to Intrusion detection.* ICSE Workshop on Software Engineering and Mobility, Canada, May 2001.

[Lange99] D. B. Lange, M. Oshima. *Seven Good Reasons for Mobile Agents.* Communications of the ACM, 1999.

[Luck03] M. Luck, P. McBurney, C. Preist. *Agent Technology: Enabling Next Generation Computing.*

[Luck00] M. Luck, M. D´Inverno. *Understanding Agent System.*

[Mahmoud04] D. Mahmoud. *Mobile Agent and Java.*

[Microsoft03] Retrieved November 2003 from Microsoft Cooperation. *http://msdn.microsoft.com/workshop/security/authcode/intro_authenticode.asp.*

[Milojicic99a] D. S. Milojicic, F. Douglis, Y. Paindaveine,R. Wheeler,S. Zhou. *Process migration.* 1999.

[Milojicic99b] D. S. Milojicic. *Mobile Agent Applications.* IEEE concurrency, 1999.

[Milojicic99c] D. S. Milojicic, F. Douglis, R. Wheeler. *Mobility, processed, computers and agents.* 1999.

[Minsky96] Y. Minsky, R. van Renesse, F. B. Schneider. *Cryptodraphic Support for Fault-Tolerant Distributed Computing.* ACM SIGOPS, pp 109-144, 1996.

[Montanari01] R. Montanari, N. Dulay, C. Stefanelli, *Flexible Security Policies for Mobile Agent Systems.* Microprocessors and Microsystems, Elsevier Science, Vol. 25, No. 2, Apr. 2001.

[Nwana96] H.S. Nwana, D. *Software Agent Overview.* Knowledge engineering overview, 1996.

[Necula98] G. C. Lee, P. Lee. *Safe, Untrusted Agent using Proof-carrying Code.* LNCS 1998.

[Opp00] Ralf Oppliger. *Security Technologies for the Internet.* Artech House Publishers, 2000.

[Pleish04] S. Pleisch, A. Schiper. *Approaches to fault-Tolerant and Transactional Mobile Agent Execution- An Algorithm View.* ACM Computing Survey, Vol. 36. 3 September 2004, pp 219-262.

[Rothermel98] K. Rothermel, M. Strasser. *A fault-tolerant protocol for providing the exactly-once property of mobile agents.* In Proc. Of the 17th IEEE symposium on reliable distributed system (SRDS'98),pp 100-108.

[Sander98] T. Sander, C. F. Tschudin. *Protecting Mobile Agents Against Malicious Hosts.* LNCS 1998.

[Schneier98] B. Schneier, J. Riordan. *Environmental Key Generation Towards Clueless Agent.* LNCS 1419, 1998.

[Tan02] H. K. Tan, L. Moreau. *Extending execution tracing for mobile code security.* SEMAS 2002.

[Tan04] H. K. Tan. *Interaction Tracing for Mobile Agent Security.* PhD Thesis, March 2004.

[Testfatson05] L. Testfatsion. *Rough definition of complex system.*

[Vigna98] G. Vigna. " Cryptographic Traces for Mobile Agents". LNCS 1419, 1999.

[Vigna04] G. Vigna. *Mobile Agents: Ten Reasons for Failure.* In the proceeding of MDM 2004,pp. 298-299, Berkeley, CA,2004.

[Westhoff99] D. Westhoff, M. Schneider, C. Unger, F. Kaderali. *Methods for protecting a mobile agent's route.* ISW 1999, LNCS 1729, 1999.

[Wilhelm99] U. G. Wilhelm, S. Staaman, L, Buttyan. *Introducing trusted Third parties to the mobile agent paradigm.* LNCS 1999.

[Yee99] B. Yee. *A Sanctuary for Mobile Agent.* Security Issues for Mobile and Distributed Object. LNCS 1999.

[Young97] A. Young, M. Yung. *Sliding Encryption: A Cryptographic tool for Mobile Agent.* Proceedings of the 4 th International Workshop on Fast Software Encryption, 1997

Refences for Chapter "Intrusion Detection"

[Amo99] E. G. Amoroso: Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Trace Back, Traps, and Response. Intrusion.Net Books, 1999.

[An80] J. P. Anderson: Computer security threat monitoring and surveillance. Technical report, J. P. Anderson Co., Ft. Washington, Pennsylvania (1980).

[Bis95] M. Bishop, D. V. Klein: Improving system security via proactive password checking. Computers & Security, 14(3):233–249, 1995.

[Blu04] C. Blundo, P. D'Arco, A. De Santis, and C. G. Hyppocrates: A New Proactive Password Checker. The Journal of Systems and Software, N. 71, pp. 163-175, 2004.

[Can98] J. Cannady: Artificial neural networks for misuse detection, Proceedings of the 1998 National Information Systems Security Conference (NISSC'98), pp. 443-456, 1998.
http://citeseer.ist.psu.edu/cannady98artificial.html

[Coh85] F. Cohen: Computer Viruses, PhD thesis, University of Southern California, 1985.

[Cri00] P. Cowan, C. Wagle, Pu, S. Beattie and J. Walpole: Buffer overflows - Attacks and defenses for the vulnerability of the decade. In Proceedings DARPA Information Survivability Conference and Exposition, pages 119–129, Hilton Head, SC, Jan. 2000.

[Den87] D. E. Denning: An intrusion-detection model. IEEE Transactions on Software Engineering, 13 (2): 222-232, February, 1987.

[Eck06] C. Eckert: IT Sicherheit - Konzepte - Verfahren - Protokolle, 4 überarbeitete Auflage, Oldenbourg, 2006.

[Ist01] S. Istvan: A comparative analysis of methods of defense against buffer overflow attacks, January 2001.
http://www.mcs.csuhayward.edu/~simon/security/boflo.html

[Ken79] R. Morris, K. Thompson: Password security: A case history, ACM, 22(11):594–597, 1979.

[Kor93] K. Ilgun: USTAT - A Real-time Intrusion Detection System for UNIX," Technical Report TRCS93-26, Computer Science Department, University of California, Santa Barbara, CA, 1993.

[Kum95] S. Kumar: Classification and Detection of Computer Intrusions, PhD Thesis, Purdue University, 1995.

[Mar01] D. J. Marchette: Computer Intrusion Detection and Network Monitoring: A statistical Viewpoint, Springer, 2001.

[Mon99] F. N. Monrose, M. K. Reiter, S. Wetzel: Password hardening based on keystroke dynamics. pages 73-82. ACM, 1999.

[Nstat] G. Vigna, R. A. Kemmerer: NetSTAT - A Network based Intrusion Detection System. Journal of Computer Security, 7(1):37–71, 1999. http://citeseer.ifi.unizh.ch/vigna99netstat.html

[Por92] P. Porras: STAT: A state Transition Analysis Tool for Intrusion Detection, Master Thesis, University of California, Santa Barbara, 1992.

[Rya98] J. Ryan, M-J. Lin, R. Miikkulainen: Intrusion Detection with Neural Networks, Advances in Neural Information Processing Systems, volume = 10, MIT Press, 1998,
http://citeseer.ist.psu.edu/ryan98intrusion.html

[RFC 791] J. Postel: Internet Protocol. RFC 791, Sept. 1981.

[RFC1750] D. Eastlake, S. Crocker, J. Schiller: Randomness Recommendations for Security, MIT December 1994.

[RFC1948] S. Bellovin: Defending Against Sequence Number Attacks, AT&T Research, May 1996.

[RFC 826] D. C. Plummer: Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48 bit Ethernet address for transmission on Ethernet hardware, November 1982.

[RFC 1519] V. Fuller, T. Li, J. Yu, K. Varadhan: Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy, September 1993.

[RFC 2460] S. Deering, R. Hinden: Internet Protocol, Version 6 (IPv6) Specification, December 1998.

[Sac93] L. Sachs: Statistische Methoden, Planung und Auswertung, Springer Verlag, 1993.

[Sch01] J. Schneiderer: Trainingsbuch SuSE Linux-Sicherheit, mitp, 2001.

[Spe03] R. Spenneberg: Intrusion Detection für Linux-Server - Mit Open Source-Tools Angriffe erkennen und analysieren.

[Spi90] R. M. Spiegel: Statistik, 2. bearbeitete Auflage, McGraw-Hill Book Company Europe, 1990.

[Wen98] W. Lee, S. Stolfo: Data mining approaches for intrusion detection, Proceedings of the 7th USENIX Security Symposium, 1998
http://citeseer.ist.psu.edu/article/lee98data.html

[Yaj00] Yan J., Y. Alan, B. Ross, A. Alasdair: The Memorability and Security of Passwords – Some Empirical Results, Technical Report No. 500, Computer Laboratory, University of Cambridge, 2000
http://citeseer.ist.psu.edu/yan00memorability.html

[Yan01] Yan, J.J: A Note on Proactive Password Checking, In Proceedings of the New Security Paradigms Workshop 2001, September 10th-13th, Cloudcroft, NM, USA pp. 127-135 ACM 2001.

[Yan04] J. Yan, Alan, Ross, Alasdair: Password Memorability and Security: Empirical Results, IEEE Security and Privacy, vol. 02, no. 5, pp. 25-31, 2004.

[Ye01] N. Ye et all: Probabilistic techniques for intrusion detection based on computer audit data. IEEE Transactions on Systems, MAN, and Cybernetics - Part A: Systems and Humans, 31 (4): 226-274, 2001.

[Ye01a] N. Ye, Q. Chen: An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. Quality and Reability Engineering International, 17, 2001.

[Ye02] N. Ye et all.: Multivariate statistical analysis of audit trails for host-based intrusion detection. IEEE Transactions on Computers, 51 (7): 810-820, 2002.

[Zav01] M. Zalewski: Strange Attractors and TCP/IP Sequence Number Analysis, BindView Corporation, 2001.
http://razor.bindview.com/publish/papers/tcpseq/print. html

Further Readings

[And01] R. Andernson: Security Engineering - A Guide to Building Dependable Distributed Systems, Wiley, 2001.

[Bej04] R. Bejtlich: The Tao of Network Security Monitoring: Beyond Intrusion Detection, 2004.

[Bis04] M. Bishop: Computer Security: Art and Science, Addison Wesley, 2004.

[Bis05] M. Bishop: Introduction to Computer Security, Addison Wesley, 2005.

[Esc98] T. Escamilla: Intrusion Detection - Network Security beyond the Firewall, Wiley 1998.

[Fei98] S. Feit: TCP/IP - Architecture, Protocols, and Implementation with IPv6 and IP Security, McGraw-Hill, 1998.

[Nor02] S. Northcutt, J. Novak: Network Intrusion Detection: An Analyst's Handbook (3rd Edition), 2002.

[Nor04] S. Northcutt, J. Novak: Network Intrusion Detection, Übersetzung der dritten Auflage des US-Standardwerkes, 2004.

[Tan02] Andrew S. Tanenbaum: Moderne Betriebssysteme, 2. bearbeitete Auflage, Pearson Studium, 2002.

Index